Hello, welcome to AP Computer Science A. This should be a fun class with lots of learning to do. To kick start your 2017-18 year I would like you to go through the following 9 lessons. At the end of the packet are some short quizzes and a test. Please go through the packet and do the quizzes and test. This should not be too onerous of a task. The reading goes quickly and the exercises are short. I will be collecting the written work when we come back to school in the fall. We will have a short test/quiz within the first 2 weeks of coming back covering the material in the summer assignment. Feel free to email me questions at oasmall@fcps.edu.

Enjoy your summer and have a good time getting off on the right foot in this AP class.

Oliver Small
Preface

You will find this book to be somewhat unusual. Most computer science texts will begin with a section on the history of computers and then with a flurry of definitions that are just “so many words” to the average student. My approach with Blue Pelican Java is to first give the student some experience upon which to hang the definitions that come later, and consequently, make them more meaningful.

This book does have a history section in Appendix S and plenty of definitions later when the student is ready for them. If you will look at Lesson 1, you will see that we go right to work and write a program the very first day. The student will not understand several things about that first program, yet he can immediately make the computer do something useful. This work ethic is typical of the remainder of the book. Rest assured that full understanding comes in time. Abraham Lincoln himself subscribed to this philosophy when he said, “Stop petting the mule, and load the wagon.”

The usual practice in most Java textbooks of introducing classes and objects alongside the fundamental concepts of primitive variable types, loops, decision structures, etc. is deferred until the student has a firm grasp of the fundamentals. Thus, the student is not overwhelmed by simultaneous introduction of OOPs (Object Oriented Programming) and the fundamentals. Once introduced, (Lesson 15), OOPs is heavily emphasized for the remainder of the book.

I fully realize that there are those who disagree with this idea of deferring the introduction of OOPs, and from their own point of view, they are right. In most cases, they teach only the very highest achieving, mature students. In those cases, I agree that it is acceptable to begin with OOPs; however, for the average student and especially for younger high school students, I feel that they need to understand the fundamentals first.

Upon first examination of this book it may not appear to be very “robust” in that there is not great depth for some of the topics. Actually the depth in there... In the Appendix. The Appendix for this book is unusually large. Here is why the book is organized this way:

The lessons are kept purposely short so as to hold down the intimidation factor. As a result, the lessons should look “doable” to the students, The in-depth material is placed in the Appendices, and references to the Appendices are made in the lessons. As an example, the split method is introduced. The split method uses regular expressions that are briefly discussed there; however, the in-depth presentation of regular expressions is placed in Appendix AC.

Unfortunately, this book does not introduce any graphics or window programming. The 57 lessons in this book can be covered in one school year, but just barely. To prepare students for the AP test and contests there is only time to cover the essentials presented in this book. Check http://www.bluepelicanjava.com for the availability of study materials for the current AP case study, updates on this book, videos for each lesson, and an inexpensive way to purchase hard-cover books.

I am often asked how to use this book. “Which lessons are really important and which can be skipped?” The answer is simple:

- Start on Lesson 1.
- Proceed at a reasonable rate. (See Appendix P for a time-line.)
- Don’t skip anything (except for perhaps Lesson 47 and Lesson 53)
- Give a simple, confidence-building quiz on each lesson. Quizzes and keys are provided in the Answer Book (available at www.bluepelicanjava.com).
- Make sure the students do the provided exercises and projects.
- Give tests at regular intervals. Tests and keys are provided in the Answer Book.

In this book you will also notice another part of my philosophy of teaching and educational material in general... I try to keep things as simple and uncluttered as possible. For example, you will find specific examples of greater numbers than long-winded explanations in this book. You won’t find many pictures and sidebars and lots of little colored side notes scattered about. Some of that type format does contain some useful information; however, I feel that it is largely distracting. Apparently more and more people are coming around to my way of thinking on this, and here is why I think so. Recall that just a few years ago that nearly all web pages looked like cobbled together ransom notes with just a profusion of colors, links, and tidbits scattered all over the page. Take a look at professional web pages today. They typically have a very neat, clean appearance... often with just a plain white background and with plenty of space between the various elements. This is good. Simple is better.

Since this textbook has a strong emphasis on preparation for the AP test and competition (computer science contests), special “contest type” problems are provided at the end of most lessons. I realize that most students will not compete and some may not even take the AP exam; however, the material is not wasted on them. Those “contest type” problems are good for the average student too, as long as they are not overwhelmed with too many problems at one sitting. I hope that you have just the optimum number of these type problems on each lesson and students won’t be burned-out by too much of a good thing.

Finally, we come to the reason for the choice of Blue Pelican Java as a name for this book. One of the (early and free) Java IDE’s available for students was BlueJ and it was the first my students used. I always thought BlueJ was an elegant name and had expressed a desire to a colleague to continue the tradition by naming the book after some other blue-colored bird. He jokingly suggested Blue Pelican, not really being serious about naming a book after this rather ungrainly, clunky bird. For the lack of an existing name for the book during development, it continued to be called Blue Pelican. If you call something by a particular name long enough, that’s its name, and so the name stuck.

I truly hope Blue Pelican Java is useful to you and that you find the experience of learning to program a rewarding one. Just remember, few things worthwhile are acquired without some sacrifice. The “sacrifice” here will be the time you invest in creating programs and trying the code suggested in these pages.

Charles E. Cook
Some of the numbered lessons below are marked with an asterisk (*). This indicates they are subjects not covered by the AP A test. All other lessons have at least "potential relevance".

<table>
<thead>
<tr>
<th>Lesson</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hello World</td>
<td>Simple use of println, rem, remarks, comments. b()ck rem.s. Project. From Me to You</td>
</tr>
<tr>
<td>2</td>
<td>Variable Types</td>
<td>String, int, double, legal names, illegal names. declaring, initializing</td>
</tr>
<tr>
<td>3</td>
<td>Simple String Operations</td>
<td>Concatenation, length, substr, toLowerCase, toUpperCase. escape sequences, backslash</td>
</tr>
<tr>
<td>4</td>
<td>Using Numeric variables</td>
<td>Assignment, ++,--$, modulus, $*$, $/$, PEMDAS increment, decrement. multiple declarations, remainder, compound operator, round-off. Project.*Cheating on Your Arithmetic Assignment</td>
</tr>
<tr>
<td>5</td>
<td>Mixed Data Types, Casting, and Constants</td>
<td>final, mixedArithmetic, casting. Project.* Mixed Result!</td>
</tr>
<tr>
<td>6</td>
<td>Math Class Methods</td>
<td>abs, pow, sqrt, ceil, floor, log, mcm, mxa, round, PT, sin, cos, tan, asin, acos, atan, toDegrees, toRadians.</td>
</tr>
<tr>
<td>7</td>
<td>*Input from the Keyboard</td>
<td>Project.* Compute This Scanner class; nextInt, nextDouble, next; next; double</td>
</tr>
<tr>
<td>8</td>
<td>boolean Type and Operators</td>
<td>AND, OR, NOT, ==, =, ==</td>
</tr>
<tr>
<td>9</td>
<td>&quot;if&quot; statement</td>
<td>equals, equalsIgnoreCase. Project... Even or Odd?</td>
</tr>
<tr>
<td>10</td>
<td>The &quot;switch&quot; Statement and chars</td>
<td>switch, default, break. char. Project.* Weight on Other Planets</td>
</tr>
<tr>
<td>11</td>
<td>The &quot;for&quot; Loop</td>
<td>Initializing, control, and stop expressions. break, infinite loops, scope, for-loop project. Project.* Name Reversal</td>
</tr>
<tr>
<td>12</td>
<td>while and do-while loops</td>
<td>Testing at top and bottom of loop. break, continue</td>
</tr>
<tr>
<td>13</td>
<td>ASCI1 and more on character</td>
<td>ASCII codes for numbers and letters. conversion from String to char. conversion from letters to String, toUpperCase, toLowerCase.</td>
</tr>
<tr>
<td>14</td>
<td>Binary, Hex, and Octal</td>
<td>Conversion between number systems. binary addition. Project.* Binary, Hex, and Octal</td>
</tr>
<tr>
<td>15</td>
<td>Classes and Objects</td>
<td>Instantiate, methods, state variables, constructor, signature. public. void. Project.* What's That Diameter?</td>
</tr>
<tr>
<td>16</td>
<td>More on Classes &amp; Objects</td>
<td>Printable methods and start variables. different lines to declare and instantiate, setting objects equal. equality of objects, assignment of objects. Project... Gas Mixture</td>
</tr>
<tr>
<td>17</td>
<td>Advanced String Methods</td>
<td>compareTo, index0f(), lastIndexOf(), charAt(). replace(). trim(Summer, reg. v-o). Project... Add 'em Up, Project.* Encryption/Decryption</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Lesson</th>
<th>Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Mayas</td>
<td>Declaring and initializing, length, parallel arrays. Out-of-bounds exception, passing an array to a method, automatic initialization, split reg. expr. Project.. Array of Hape</td>
</tr>
<tr>
<td>19</td>
<td>Advanced Array Concepts</td>
<td>Arrays of objects, comparison of array values, null pointer exception, different reference to same array, arraycopy, toCharArray, logical vs physical size, Arrays class, sort, binarySearch. equals, fill. importing command line arguments, enhanced for-loop. Project.* Sorting a String Array. Project.* Two Orders for the Price of One</td>
</tr>
<tr>
<td>20</td>
<td>Static Methods and State Variables</td>
<td>Class methods and variables, static constants static imports. Project. How Far To The Line?</td>
</tr>
<tr>
<td>21</td>
<td>Wrapper Classes</td>
<td>Converting primitives to objects and vice versa</td>
</tr>
<tr>
<td>22</td>
<td>More on Wrapper Classes</td>
<td>parseInt, parseDouble, toHexString, toOctalString, toString, toString(), valueOf</td>
</tr>
<tr>
<td>23</td>
<td>*Input from 11 Disk File</td>
<td>Scanner, File, IEnumerator, readLines, Project for Reading Files, close, Project. Reading Files</td>
</tr>
<tr>
<td>24</td>
<td>*Processing File Input with Scanner</td>
<td>Processing text coded numbers, usingparseInt and parseDouble, parsing and manipulating text. Project.* Get Rid Of That Plus Sign! Project.* Student Averages</td>
</tr>
<tr>
<td>25</td>
<td>*Writing to a Text File</td>
<td>FileWriter, PrintWriter, print, println. appending to the end of a file, close. Project. Write Student Averages</td>
</tr>
<tr>
<td>26</td>
<td>*Formatting (rounding off)</td>
<td>NumberFormat, formatting numbers, currency, and percent. Format class, print. Project... BaseClass. Project. L. Gymnastics</td>
</tr>
<tr>
<td>27</td>
<td>*Bitwise operators</td>
<td>Bitwise-AND, OR, exclusive-OR, and NOT. Negative numbers, sign bit, most significant bit</td>
</tr>
<tr>
<td>28</td>
<td>*AdvancedBitwise Operations</td>
<td>Shift left and right. &lt;&lt;, &gt;&gt;&gt;, &gt;&gt;&gt;&gt;, preservation of sign, short-circuit, precedence. Negative numbers, sign bit, most significant bit. Project.*</td>
</tr>
<tr>
<td>29</td>
<td>SRandom Numbers</td>
<td>neA1Double, nextInt, Monte Carlo, simulations. Project.* Monte Carlo Technique</td>
</tr>
<tr>
<td>30</td>
<td>*StringBuffer Class</td>
<td>append, toString, substring, length, setCharAt, delete, deleteCharAt, insert, charAt. Project... Concatenates Gone Wild</td>
</tr>
<tr>
<td>31</td>
<td>*Boolean Algebra and DeMorgan's Theorem</td>
<td>OR, AND, truth table. Project.*</td>
</tr>
<tr>
<td>32</td>
<td>*Selection Operator</td>
<td>7-syntax</td>
</tr>
<tr>
<td>33</td>
<td>*Passing by Value and by Reference</td>
<td>Arrays, primitives. objects. references. Project.* Pass the Gravy, Please</td>
</tr>
<tr>
<td>34</td>
<td>*isLoggedIn-Dimensional Arrays</td>
<td>Subscribed variables. matrix. initializing. Arrays class. Project.* Math. Multiplication, Project.* Matrix Multiplication with File Input</td>
</tr>
<tr>
<td>35</td>
<td>Inheritance</td>
<td>Superclass, subclass. base class. derived class, abstract, final. overriding. shadowing. cosmic superclass, instanceof. Object. this. super</td>
</tr>
</tbody>
</table>
Comparables/Comparators

Recursion, Factorial, Fibonacci series, Project...

Interfaces

Checked, unchecked, try, catch, finally, throw, throws, Project...

Implementation perspective, objective perspective. instance, polymorphism, realizes, implements. Project...

Comparable/Comparator

Linear Function

Sequential search, binary search

Complexity Analysis (Big O)

Factorial, Fibonacci series, Project...

Recursion

selection, insertion, quick, & merge sorts, partition, big O chart. Project...

Multiple-Key Sorting

Array, List, LinkedList, Vector

Advantages, disadvantages, Project...

Big Buck in the Bank

Arrays, List, LinkedList, Set interface, iterators, Project...

HashSet, TreeSet

Set interface, Iterators

Singly Linked list, doubly linked list

Sorting Routines

Selection, Insertion, Quick

List Interface

ArrayList, LinkedList, Vector

Array, List, LinkedList, Set interface, iterators, Project...

HashSet, TreeSet

HashSet, TreeSet

Set interface, Iterators

Singly Linked list, doubly linked list

Interfaces

Checked, unchecked, try, catch, finally, throw, throws, Project...

implementation perspective, objective perspective. instance, polymorphism, realizes, implements, Project...

Complexity Analysis (Big O)

Recursion

Factorial, Fibonacci series, Project...

check, Project...

sorting

Selection, Insertion, Quick

List Interface

ArrayList, LinkedList, Vector

HashSet, TreeSet

HashSet, TreeSet

Set interface, Iterators

Binary Search

Arrays, sorts, ArraysBinarySearch

Project...

Binary search with Objects

Project...

Stack class

Binary Search Trees

Binary search trees, preorder, inorder, postorder, and in level traversals. Expression trees. Project...

BST

find method

Priority Queues

Highest

Priority queue, array implementation...

Project ...

This is a plain text representation of the document as if you were reading it naturally.
<table>
<thead>
<tr>
<th>Table of Contents - 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>#26 Golden Nugget of Wisdom #26</td>
</tr>
<tr>
<td>#27 Golden Nugget of Wisdom #27</td>
</tr>
<tr>
<td>#28 Golden Nugget of Wisdom #28</td>
</tr>
<tr>
<td>#29 Golden Nugget of Wisdom #29</td>
</tr>
<tr>
<td>#30 Golden Nugget of Wisdom #30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Appendix A</th>
<th>Key Words</th>
<th>Reserved words that are part of Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix B</td>
<td>Escape Sequences</td>
<td></td>
</tr>
<tr>
<td>Appendix C</td>
<td>Primitive Data Types</td>
<td>byte, short, int, long, float, double, char, boolean</td>
</tr>
<tr>
<td>Appendix D</td>
<td>ASCII Codes</td>
<td>Decimals, hex, octal, and bin equivalents</td>
</tr>
<tr>
<td>Appendix E</td>
<td>Saving Text Files</td>
<td>Windows settings, Notepad, WordPad</td>
</tr>
<tr>
<td>Appendix F</td>
<td>Two's Complement Notation</td>
<td>Storage methods</td>
</tr>
<tr>
<td>Appendix G</td>
<td>Operator Precedence</td>
<td>Negative numbers, invert, ones' complement, ten's complement, odometer, num, sign bit</td>
</tr>
<tr>
<td>Appendix H</td>
<td>Creating Packages and Importing Classes</td>
<td>Order of operations</td>
</tr>
<tr>
<td>Appendix I</td>
<td>Types, Access Specifiers, and Interfaces</td>
<td>Importing, package, wildcard, 64steps to create a package, classpath variable</td>
</tr>
<tr>
<td>Appendix J</td>
<td>Exception Classes</td>
<td>Scope of variables, exception</td>
</tr>
<tr>
<td>Appendix K</td>
<td>All Essay on Interfaces</td>
<td>A list of some checked and unchecked exceptions</td>
</tr>
<tr>
<td>Appendix L</td>
<td>Input from the Keyboard</td>
<td>Down to earth explanation of interfaces</td>
</tr>
<tr>
<td>Appendix M</td>
<td>Using the BlueJ Programming Environment</td>
<td>BufferedReader, InputStreamReader</td>
</tr>
<tr>
<td>Appendix N</td>
<td>Using the JCreator Programming Environment</td>
<td></td>
</tr>
<tr>
<td>Appendix O</td>
<td>Intermediate for Lessons</td>
<td>Time allocation for each lesson</td>
</tr>
<tr>
<td>Appendix P</td>
<td>AP(A) Correlation</td>
<td>Page number correlation</td>
</tr>
<tr>
<td>Appendix Q</td>
<td>Texas TEKS/STAks Correlation</td>
<td>Page number correlation to TEKS</td>
</tr>
<tr>
<td>Appendix R</td>
<td>History of Computers</td>
<td>Pascal, Babbage, ENIAC, operating systems, MITS Altair, TRS-80, Apple, IBM pc, disk storage, key punch cards</td>
</tr>
<tr>
<td>Appendix T</td>
<td>Viruses</td>
<td>What a virus is, how they are spread, types, protection, ethics, and etiquette</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table of Contents - 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix U</td>
</tr>
<tr>
<td>Appendix V</td>
</tr>
<tr>
<td>Appendix W</td>
</tr>
<tr>
<td>Appendix X</td>
</tr>
<tr>
<td>Appendix Y</td>
</tr>
<tr>
<td>Appendix Z</td>
</tr>
<tr>
<td>Appendix AA</td>
</tr>
<tr>
<td>Appendix AB</td>
</tr>
<tr>
<td>Appendix AC</td>
</tr>
<tr>
<td>Appendix AD</td>
</tr>
<tr>
<td>Appendix AE</td>
</tr>
<tr>
<td>Appendix AF</td>
</tr>
<tr>
<td>Appendix AG</td>
</tr>
<tr>
<td>Appendix AH</td>
</tr>
</tbody>
</table>
Lesson 1...Hello World

Program Skeleton:
Enter the following program skeleton, compile (preprocess), and then run (execute).
Your instructor may have you give it a specific project name; otherwise, call the project Lesson.

If you do not know how to enter and execute a program, ask your instructor, or use the appendices in this book for two of the more popular programming environments. See Appendix N for the BlueJ environment and Appendix O for the JCreator environment.

Two printlns for the price of one:
Next, modify your program so that the main method looks as follows:

```java
public static void main(String args[]) {
    System.out.println("Hello world");
    System.out.println("Hello again");
}
```

At this point don’t worry about what any of this means. It’s just something we must do every time. Soon we will learn the meaning of all of this. For now it’s just the skeleton that we need for a program.

Adding some meaningful code:
Now, let’s add some meaningful code inside the main method. (Notice this word: method. We will constantly refer to methods throughout this course.) We will also add a remark.

```java
public class Tester //We could put any name here besides Tester
{
    public static void main(String args[]) {
        System.out.println("Hello world");
    }
}
```

Remarks:
Notice the rem (remark) above that starts with //: You can put remarks anywhere in the program without it affecting program operation. Remarks are also called comments or notes.

Printing:
`System.out.println("Hello world");` is how we get the computer to print out something. Notice the trailing semicolon. Most lines of code are required to end in a semicolon;

Now try putting in some other things in the println parenthesis above. Each time recompile and run the program:

1. "Peter Piper picked a peck of pickled peppers."
2. "I like computer science."
3. 25/5
4. 4.70445902
5. -13 * 15986

Two printlns for the price of one:
Next, modify your program so that the main method looks as follows:

```java
public static void main(String args[]) {
    System.out.println("Hello world");
    System.out.println("Hello again");
}
```

Run this and note that it prints:
Hello world
Hello again

Printing "Sideways."
Now remove the In from the first println as follows:

```java
public static void main(String args[]) {
    System.out.print("Hello world");
    System.out.println("Hello again");
}
```

Run this and note that it prints:
Hello world Hello again

Here are the rules concerning println and print:
- `System.out.println()` completes printing on the current line and pulls the print position down to the next line where any subsequent printing continues.
- `System.out.print()` prints on the current line and stops there. Any subsequent printing continues from that point.

An in-depth look at rems:
Let’s take a further look at rems. Consider the following program (class) in which we wish to document ourselves as the programmer, the date of creation, and our school:

```java
public class Tester //We could put any name here besides Tester
{
    public static void main(String args[]) {
        System.out.println("Hello again");
    }
}
```

1. "Peter Piper picked a peck of pickled peppers."
2. "I like computer science."
Block rem:

It can get a little tedious putting the double slash rem-indicator in front of each line, especially if we have quite a few remark lines. In this case we can "block rem" all the comment lines as follows:

```java
public class Tester {
    /*Programmer: Kosmo Kramer
    Date created: Sept 34, 492
    School: Charles Manson Junior High; Berkley, Ca*/
    public static void main(String args0)
    {
        System.out.println("Hello again");
    }
}
```

Notice we use /* to indicate the start of the block and */ for the end. Everything between these two symbols is considered to be a remark and will be ignored by the computer when compiling and running.

Lesson 2....Variable Types (String, int, double)

Three variable types:

(A good way to learn the following points is to modify the code of the "Hello World" program according to the suggestions below.)

1. `String` ....used to store things in quotes....like "Hello world"
   Sample code:
   ```java
   public static void main(String args0)
   {
       String s = "Hello cruel world";
       System.out.println(s);
   }
   ``

2. `int` ....used to store integers (positive or negative)
   Sample code:
   ```java
   public static void main(String args0)
   {
       int age = 59;
       System.out.println(age);
   }
   ```

With the advent of Java 7.0, numbers that were previously very awkward and difficult to read can now be entered with underscore separators.

For example,

```java
int bigNum = 1389488882; can now be entered as
int bigNum = 1_389_488_882;
```

Unfortunately, commas still cannot be used as separators.

3. `double` ....used to store "floating point" numbers (decimal fractions). `double` means "double precision".
   Sample code:
   ```java
   public static void main(String args[])
   {
       double d = -137.8036;
       System.out.println(d);
   }
   ``

   `d = 1.45667E23: //Scientific notation means 1.45667 X 10^23`

Declaring and initializing:

When we say something like

```java
double x = 1.6;
```
we are really doing two things at once. We are declaring x to be of type double and we are initializing x to the value of 1.6. All this can also be done in two lines of code (as shown below) instead of one if desired:

double x; //this declares x to be of type double  
x = 1.6;  //this initializes x to a value of 1.6

What's legal and what's not:
int arws = 47.4;  //illegal, won't compile since a decimal number cannot "fit" into an integer variable.

double d = 103;  //legal...same as saying the decimal number 103.0

Rules for variable names:
Variable names must begin with a letter (or an underscore _ character) and cannot contain spaces. The only "punctuation" character permissible inside the name is the underscore ("_"). Variable names cannot be one of the reserved words (key words...see Appendix A) that are part of the Java language.

<table>
<thead>
<tr>
<th>Legal names</th>
<th>Illegal names</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agro</td>
<td>139</td>
</tr>
<tr>
<td>d31</td>
<td>fut Ono</td>
</tr>
<tr>
<td>hoppergee</td>
<td>class</td>
</tr>
<tr>
<td>hopper_gee</td>
<td>slow.Sally</td>
</tr>
<tr>
<td>largeArea</td>
<td>double</td>
</tr>
<tr>
<td>goldNugget</td>
<td>gold.Nugget</td>
</tr>
<tr>
<td>hopper-gee</td>
<td>hopper-gee</td>
</tr>
</tbody>
</table>

Variable naming conventions:
It's traditional (although not a hard and fast rule) for variable names to start with a lower case letter. If a variable name consists of multiple words, combine them in one of two ways:

bigValue ... jam everything together. First word begins with a small letter and subsequent words begin with a capital.
big_value...separate words with an underscore.

Exercise on Lesson 2

1. What are the three main types of variables used in Java and what are they used to store?

2. What type of variable would you use to store your name?

3. What type of variable would you use to store the square root of 2?

4. What type of variable would you use to store your age?

5. Write a single line of code that will create a double precision variable called p and store 1.921X 10^-16 in it.

6. Write a single line of code that will create an integer variable called i and store 407 in it.

7. Write a single line of code that will create a String variable called my_name and store your name in it.

8. Write a line of code that will declare the variable count to be of type int. Don't initialize.

9. Write a line of code that initializes the double precision variable bank.Balance to 136.05. Assume this variable has already been declared.

10. Which of the following are legal variable names?
    scooter13  139_scooter  homer-5  mary public doubled double ab c

11. Which of the following is the most acceptable way of naming a variable. Multiple answers are possible.
    a. GroovyDude
    b. GROOVYDUDE
    c. groovyDude
    d. Groovydude
    e. groovy_dude
    f. groovydude

12. Comment on the legality of the following two lines of code.
    double dist = 1003;
    int alt = 1493.86;
    int num = 1_2_3,
Lesson 3...Simple String Operations

In this lesson we will learn just a few of the things we can do with Strings.

Concatenation:
First and foremost is concatenation. We use the plus sign, +, to do this. For example:

```java
String mm = "Hello";
String nx = "good buddy";
String c = mm + nx;
System.out.println(c); //prints Hellogood buddy ..notice no space between o & g
```

The above code could also have been done in the following way:

```java
String mm = "Hello";
String nx = "good buddy";
System.out.println(mm + " " + nx); //prints Hello good buddy ...notice the space
```

We could also do it this way:

```java
String bismark = "Dude, where's MY car?";
System.out.println(bismark.toLowerCase()); //prints dude, where's my l"ar?
```

```java
toUpperCase
```
converts all characters to upper case (capital letters)

```java
System.out.println("Dude, where's My car?".toUpperCase());
//prints DUDE, WHERE'S MY CAR?
```

Note: length, substring, toLowerCase, andtoUpperCase are all methods of the String class. There are other methods we will learn later.

Concatenating a String and a numeric:
It is possible to concatenate a String with a numeric variable as follows:

```java
System.out.println("Hello" good buddy"); // prints Hello good buddy
int x = 27;
String s = "Was haben wir gemacht?"; //German for "What have we done?"
String combo = s +" " +x;
```

The length method:
Use the length() method to find the number of characters in a String.

```java
String theName = "Donald Duck";
int len = theName.length();
System.out.println(len); //prints 11 ..notice the space gets counted
```

```java
System.out.println(combo); //prints Was haben wir gemacht? 27
```

Escape sequences:
How do we force a quote character ("') to printout .... or, to be part of a String. Use the escape sequence, \, to print the following (note escape sequences always start with the \ character...see Appendix B for more on escape sequences):

```java
Right now we don't see much value in this length thing...just wait!
```

A piece of a String substring:
We can pick out a piece of a String ...substring

```java
String myPet = "Sparky the dog";
String smallPart = myPet.substring(4);
System.out.println(smallPart); //prints ky the dog
```

Why do we get this result? The various characters in a String are numbered starting on the left with 0. These numbers are called indices. (Notice the spaces are numbered too.)

```
Sparky the dog. 0 1 2 3 4 5 6 7 8 9 10 11 12 13
```

A more useful form of substring:
But wait! There's another way to use substring

```java
String myPet = "Sparky the dog";
String smallPart = myPet.substring(4, 12);
System.out.println(smallPart); //prints ky the dog
```

How do we get ky the !! Start at k, the 4th index, as before. Go out to the 12th.index, ..'o' in this case and pull back one notch. That means the last letter is d.
Exercise on Lesson 3

1. Write code in which a String variable s contains "The number of rabbits is". An integer variable argh has a value of 129. Concatenate these variables into a String called report. Then print report. The printout should yield: The number of rabbits is 129.

Note that we want a period to print after the 9.

2. What is the output of: System.out.println(p.toUpperCase()); if p = "Groovy Dude"?

3. Write code that will assign the value of "Computer Science is for nerds" to the String variable g. Then have it print this String with nothing but "small" letters.

4. What will be the value of c?
   String c;
   String m = "The Gettysburg Address";
   c = m.substring(4);

5. What will be the value of c?
   String b = "Four score and seven years ago,";
   String c = b.substring(7, 12);

6. What is the value of count?
   int count;
   String s = "Surface tension";
   count = s.length();

7. Write code that will look at the number of characters in String m "Look here!"; and then print "Look here!" has 10 characters.

   Use the length() method to print the 10...you must also force the two quotes to print.

8. How would you print the following?
   All "good" men should come to the aid of their country.

9. Write code that will produce the following printout Using only a println().

    Hello
    Hello again

10. Write code that will produce the following printout

    A backslash looks like this \\,...right?

11. What is output by the following?

    String pq = "Eddie Haskel";
    int hm = pq.length();
    String ed = pq.substring(hm - 4);
    System.out.println(ed);

12. Which character is at the 5th index in the String "Herman Munster"?

Project... Name that Celebrity

Create a new project called NameThatCelebrity in which only partially recognizable names of celebrities are to be produced. In a real implementation of this game, the idea is for a contestant to be able to guess the real name of the celebrity after the first two and last three letters are dropped from the name. We have been given the task of testing the feasibility of this idea by producing the following printout:

<table>
<thead>
<tr>
<th>All \a\l\a\l A</th>
</tr>
</thead>
<tbody>
<tr>
<td>John \n W a</td>
</tr>
<tr>
<td>Greg \r ey P</td>
</tr>
</tbody>
</table>

Begin your code within the main method as follows:

String sl = "Allan Alda";
String s2 = "John Wayne";
String s3 = "Gregory Peck";

Apply the length and substring methods to these Strings to produce the above printout.
Lesson 4...Using Numeric Variables

The assignment operator:
The assignment operator is the standard equal sign (=) and is used to "assign" a value to a variable.

```java
int i = 3; // Ok...assign the value 3 to variable i.
```

The increment operator is `++`, and it means to add one. The decrement operator is `--`, and it means to subtract one:

```java
x++; means the same as x = x + 1;
x--; means the same as x = x - 1;
x++ is the same as ++x (the ++ can be on either side of x)
x-- is the same as --x (the -- can be on either side of x)
```

Multiple declarations:
It is possible to declare several variables on one line:

```java
double d, mad, puma; // the variables are only declared
double x = 31.2, m = 37.09, p = 43.917; // x, m, & p declared and initialized
```

Fundamental arithmetic operations:
The basic arithmetic operations are `+`, `-`, `*` (multiplication), `/` (division), and `%` (modulus).

Modulus is the strange one. For example, `System.out.println(5%3);` will print 2. This is because when 5 is divided by 3, the remainder is 2. Modulus gives the remainder. Modulus also handles negatives. The answer to a % b always has the same sign as a. The sign of b is ignored.

PEMDAS:
The algebra rule, PEMDAS, applies to computer computations as well. (PEMDAS stands for the order in which numeric operations are done. P = parenthesis, E = exponents, M = multiply, D = divide, A = add, S = subtract. Actually, M and D have equal precedence, as do A and S. For equal precedence operation, proceed from left to right. A mnemonic for PEMDAS is "Please excuse my dear Aunt Sally"...See Appendix H for the precedence of all operators.)

```java
System.out.println(5 + 3 * 4 - 7); // 10
System.out.println(8 - 5*2 + (5 - 6) * 3); // 9
```

Not the same as in Algebra:

An unusual assignment. Consider the following:

```java
count = count + 3; // this is illegal in algebra; however, in computer science it means the old count + 3.
```

```java
int count = 15;
count = count + 3;
```

```
System.out.println(count); // 18
```

Increment and Decrement:
The increment operator is `++`, and it means to add one. The decrement operator is `--`, and it means to subtract one:

```java
x++; means the same as x = x + 1;
x--; means the same as x = x - 1;
x++ is the same as ++x (the ++ can be on either side of x)
x-- is the same as --x (the -- can be on either side of x)
```

```java
int y = 3;
y++;
System.out.println(y); // 4
```

Compound operators:

```java
a. x += 3; 7 x = x + 3;
b. x -= y - 2; 7 x = x - (y - 2);
c. z*=46; 7 z = z * 46;
d. /=2 7 p = p / (x - z);
e. %=2 7 j = j % 2;
```

Code Examples:

```java
System.out.println(5 + 3 * 4 - 7); // 10
System.out.println(8 - 5*2 + (5 - 6) * 3); // 9
```

```
int g = 409;
g += 5;
System.out.println(g); // 414
```

```java
double d = 20.3;
double m = 10.0;
m*=d-1;
System.out.println(m); // 193.0
```

The whole truth:

Actually, the full truth was not told above concerning `x++`. It does not always have the same effect as does `++x`. Likewise, `x-` does not always have the same effect as does `-x`.

```java
x++ increments x after it is used in the statement.
++x increments x before it is used in the statement.
```
Similarly,

- \( x-- \) decrements \( x \) after it is used in the statement.
- \( --x \) decrements \( x \) before it is used in the statement.

**Code Examples**

```java
int q = 78;
int p = 2 + q++; // p = 30, q = 78
System.out.println("p = " + p + ", q = " + q);
```

```java
int q = 78;
int p = ++q + 2; // p = 80, q = 79
System.out.println("p = " + p + ", q = " + q);
```

**Integer division truncation:**

When dividing two integers, the fractional part is truncated (thrown away) as illustrated by the following:

```java
int x = 5;
int y = 2;
System.out.println(x / y); // Both x and y are integers so the "real" answer of 2.5
// has the fractional part thrown away to give 2
```

**Exercise on Lesson 4**

Unless otherwise directed in the following problems, state what is printed. Some of these problems may have incorrect syntax and in those cases you should answer that the code would not compile.

1. int h = 103;
   int p = 5;
   System.out.println("\+h + p");
   System.out.println(h);

2. Give three code examples of how to increment the integer \( j \) by \( L \).

3. double def;
   double f = 1992.37;
   def = f;
   System.out.println(def);

4. Write a single line of code that will print the integer variable \( zulu \) and then decrement its value by 1.

5. int a = 100;
   int b = 200;
   b = a;
   System.out.println(b + 7);

6. Write a single line of code that uses the compound operator, -=, to subtract \( p-30 \) from the integer value \( v \) and store the result back in \( v \).

7. Write a single line of code that does the same thing as \#6 but without using -=.

8. int p = 40;
   int q = 4;
   System.out.println(2 + 8 * q / 2 - p);

9. int sd = 12;
   int x = 4;
   System.out.println( sd%(+x) );
   System.out.println(x);

10. int g;
    3 = g;
    System.out.println(++g*79); // What is the result?

11. On a single line of code declare \( m, h, \) and \( f \) to be double and on that same line initialize them all to be 3.14.

12. On a single line of code declare \( x, y, \) and \( z \) all to be of integer type.

13. int m = 36;
    int j = 5;
    m = m / j; // new m is old m divided by j
    System.out.println(m);
    What's printed?

14. System.out.println(3/4 + 5*2/33.3 + 8*3);
    What's printed?
15. What is the assignment operator?

16. Write a statement that stores the remainder of dividing the variable $i$ by $j$ in a variable named $k$.

17. `int j = 2;
    System.out.println(7%3 + j++ + G -2);`

18. Show three different ways to decrement the variable $j$.

---

**Lesson 5: Mixed Data Types, Casting, and Constants**

So far we have looked mostly at simple cases in which all the numbers involved in a calculation were either all integers or all doubles. Here, we will see what happens when we mix these types in calculations.

Java doesn't like to lose data:

Here is an important principle to remember: Java will not normally store information in a variable if in doing so it would lose information. Consider the following two examples:

1. An example of when we would lose information:
   
   ```java
   int d = 29.78;
   int i = d; //won't compile since i is an integer and it would have to chop-off the .78 and store just 29 in i...thus, it would lose information.
   ```

   There is a way to make the above code work by forcing compilation and therefore store 29.78 being "stored" in i as follows (actually, just 29 is stored since i can only hold integers):

   ```java
   int i = (int)d;  //(int) "casts" d as an integer. It converts d to integer form.
   ```

2. An example of when we would not lose information:

   ```java
   int j = 105;
   double d = j; //legal, because no information is lost by storing 105 in the double variable d.
   ```

---

**Project: Cheating on Your Arithmetic Assignment**

Create a new project called `ArithmeticAssignment` with a class called `Tester` that will calculate and print the results of the following arithmetic problems:

1. $79 + 3 \times (4 + 82 - 68) - 7 + 19$
2. $(179 + 21 + 10) / 7 + 181$
3. $10389 \times 56 + 11 + 2246$

The printout should look like the following:

1. $79 + 3 \times (4 + 82 - 68) - 7 + 19 = 145$
2. $(179 + 21 + 10) / 7 + 181 = 211$
3. $10389 \times 56 + 11 + 2246 = 6401870$

---

```
10389 * 56 * 11 + 2246
10389 * 56 * 11 + 2246 = 6401870
```
Constants:

Constants follow all the rules of variables; however, once initialized, they cannot be changed. Use the keyword `final` to indicate a constant. Conventionally, constant names have all capital letters. The rules for legal constant names are the same as for variable names. Following is an example of a constant:

```java
final double PI = 3.14159;
```

The following illustrates that constants can't be changed:

```java
final double PI = 3.14159;
PI = 3.7789; //illegal
```

When in a method, constants may be initialized after they are declared.

```java
final double PI; //legal
PI = 3.14159;
```

Constants can also be of type `String`, `int`, and other types.

```java
final String NAME = "Peewee Herman";
final int LUNCH_COUNT = 122;
```

The real truth about compound operators:

In the previous lesson we learned that the compound operator `expression+x;` was equivalent to `expression+=x;` . Actually, for all compound operators there is also an implied cast to the type of `expression`. For example, if `expression` is of type `int`, the real meaning of `expression+=x;` is:

```java
expression = (int)expression + x;
```

Project... Mixed Results

Create a new project called `MixedResults` with a class called `Tester`. Within the `main` method of `Tester` you will eventually printout the result of the following problems. However, you should first calculate by hand what you expect the answers to be. For example, in the parenthesis of the first problem, you should realize that strictly integer arithmetic is taking place that results in a value of 0 for the parenthesis.

```java
double d1 = 37.9; //Initialize these variables at the top of your program
double d2 = 1004.128;
int i1 = 12;
int i2 = 18;
```

Exercise on Lesson 5

Unless otherwise instructed in the following problems, state what gets printed.

1. Write code that will create a constant `E` that's equal to 2.718.

```java
final double E = 2.718;
```

2. Write the simplest type constant that sets the number of students, `NUM_STUDENTS`, to 236.

```java
final int NUM_STUDENTS = 236;
```

3. What's wrong, if anything, with the following code in the `main` method?

```java
final double Area;
Area = 203.49;
```

4. int cnt = 27.2;
   System.out.println(cnt);
   What's printed?

5. double d = 78.1;
   int fg = (int)d;
   System.out.println(fg);
   What's printed?

6. Is `double f4 = 22;` legal?

7. The following code stores a 20 in the variable):

```java
double j = 61/3;
```

   What small change can you make to this single line of code to make it produce the "real" answer to the division?

8. System.out.println((double)(90/9));
Lesson 6...••.Methods of the Math Class

One of the most useful methods of the Math class is sqrt()...which means square root For example, if we want to take the square root of 17 and store the result in p, do the following:

double p = Math.sqrt(17);

Notice that we must store the result in a double in this case. We must store in a double since square roots usually don't come out even.

Signature of a method: Below we will give the description of some methods of the Math class...along with the signatures of the methods. First, however, let's explain the meaning of signature (also called a method declaration). Consider the signature of the sqrt() method:

double sqrt( double x )

type returned  method name  type of parameter we send to the method

Method                     Signature         Description
ab'                      int abs(int x)     Returns the absolute value of x
ab'                      double abs(double x) Returns the absolute value of x
pow                       double pow(double b, double e) Returns b raised to the e power
floor                     double floor(double x) Returns lowest whole number from x
floor                     double floor(double x, double x) Returns next lowest whole number from x
min                       double min(double a, double b) Returns the smaller of a and b
max                       double max(double a, double b) Returns the larger of a and b
min                       int min(int a, int b) Returns the smaller of a and b
max                       int max(int a, int b) Returns the larger of a and b

(For both min and max there are also versions that both accept and return types float, short, and long. See Appendix C for more on these three data types.)

random                   long random()      Returns random double (range 0.5 < 1)
round                    long round(double x) Returns x rounded to nearest whole number
PI                        double PI          Returns 3.14159625...

Now, we offer examples of each (most of these you can do on a calculator for verification):

1. double d = -379.22;
   System.out.println( Math.abs(d) );   //379.22
2. double b = 4201;
   double e = 3.7281;
   System.out.println( Math.sqrt(b, e) );   //1126831.027
3. double d = 2034.56;
   System.out.println( Math.ceil(d) );   //2.0
4. double d = 1.4;
   System.out.println( Math.ceil(d) );   //2.0

5. System.out.println(4 + 6/4 + 5 * 3 - 3);
6. int p = 3;
   double d = 10.3;
   int j = (int)5.9;
   System.out.println(p * p * d - 3 * j);

7. int dividend = 12, divisor = 4, quotient = 0, remainder = 0;
   int dividend2 = 13, divisor2 = 3, quotient2 = 0, remainder2 = 0;
   quotient = dividend/divisor;
   remainder = dividend % divisor;
   quotient2 = dividend2/divisor2;
   remainder2 = dividend2 % divisor2;
   System.out.println(quotient);

8. System.out.println(remainder);
9. System.out.println(quotient2);
10. System.out.println(remainder2);

11. Write a line of code in which you divide the double precision number d by an integer variable called i. Type cast the double so that strictly integer division is done. Store the result in another integer.

12. Suppose we have a line of code that says

    final String M = "ugg";

   Later in the same program, would it be permissible to say the following?

    M = "row";

13. Is the following code legal? If so, what is printed? If not, why?

    int k = 7;
    k*=-5;
    System.out.println(k);
5. double d = -1.6;
   System.out.println( Math.ceil(d) ); // -1.0

6. double d = 1.4;
   System.out.println( Math.floor(d) ); // 1.0

7. double d = -1.6;
   System.out.println( Math.floor(d) ); // -2.0

8. double x = 2038.5;
   System.out.println( Math.min(x,y) ); // -8999.0

9. double x = 2038.5;
   System.out.println( Math.max(x,y) ); // 2038.5

10. double x = 148.2;
    System.out.println( Math.round(x) ); // 148

   double x = -148.2;
   System.out.println( Math.round(x) ); // -148

11. double x = 148.7;
    System.out.println( Math.round(x) ); // 149

   double x = -148.7;
   System.out.println( Math.round(x) ); // -149

12. System.out.println( Math.PI ); // 3.14159265...

Advanced Math methods:
Below are some additional Math methods that advanced math students will find useful:

<table>
<thead>
<tr>
<th>Method</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>log</td>
<td>double log(double x)</td>
<td>Returns log base e of x</td>
</tr>
<tr>
<td>sin</td>
<td>double sin(double a)</td>
<td>Returns the sine of angle a... a is in rad</td>
</tr>
<tr>
<td>cos</td>
<td>double cos(double a)</td>
<td>Returns the cosine of angle a... a is in rad</td>
</tr>
<tr>
<td>tan</td>
<td>double tan(double a)</td>
<td>Returns the tangent of angle a... a is in rad</td>
</tr>
<tr>
<td>asin</td>
<td>double asin(double x)</td>
<td>Returns arcsine of x... in range -PI/2 to PI/2</td>
</tr>
<tr>
<td>acos</td>
<td>double acos(double x)</td>
<td>Returns arccosine of x... in range 0 to PI</td>
</tr>
<tr>
<td>atan</td>
<td>double atan(double x)</td>
<td>Returns arctan of x in range -PI to PI</td>
</tr>
<tr>
<td>toDegrees</td>
<td>double toDegrees(double angRad)</td>
<td>Converts radians into degrees</td>
</tr>
<tr>
<td>toRadians</td>
<td>double toRadians(double angDeg)</td>
<td>Converts degrees into radians</td>
</tr>
</tbody>
</table>

Exercise on Lesson 6

1. Write code that will take the square root of x and store the result in y.

2. Write code that will multiply the value of the integer f times the absolute value of the integer m and then store the result in the integer k.

3. Is the following legal? If not, what would you do to make it legal?
   ```java
   int k = Math.abs(-127.5);
   ```

4. Write a statement that will print the result of $2^{0.5}$.

5. System.out.println( Math.ceil(-157.2) );

6. System.out.println( Math.floor(-157.2) );

7. System.out.println( Math.ceil(-157.2) );

8. System.out.println( Math.floor(-157.2) );

9. System.out.println( Math.ceil(-157.2) );

10. System.out.println( Math.floor(-157.2) );

11. System.out.println( Math.ceil(-157.2) );
Lesson 7. Input from the Keyboard

We will consider how to input from the keyboard the three data types: int, double, and String.

Inputting an integer:
Use the nextInt method to input an integer from the keyboard:
```java
import java.io.*; // see "Imports necessary" on next page
import java.util.*;
public class Tester
{
    public static void main(String args[])
    {
        Scanner kbReader = new Scanner(System.in); // see "Mysterious objects" on next page
        System.out.print("Enter your integer here. "); // enter 3001
        int i = kbReader.nextInt();
        System.out.println(3 * i); // prints 9003
    }
}
```

Inputting a double:
Use the nextDouble method to input a double from the keyboard:
```java
import java.io.*;
import java.util.*;
public class Tester
{
    public static void main(String args[])
    {
        Scanner kbReader = new Scanner(System.in);
        System.out.print("Enter your decimal number here. "); // 10005
        double d = kbReader.nextDouble();
        System.out.println(3 * d); // prints 30015
    }
}
```

Inputting a String:
Use the next method to input a String from the keyboard:
```java
import java.io.*;
import java.util.*;
public class Tester
{
    public static void main(String args[])
    {
        Scanner kbReader = new Scanner(System.in);
        System.out.print("Enter your String here. "); // Enter One Two
        String s = kbReader.next(); // inputs up to first white space
        System.out.println("This is the first part of the String:") + s);
        s = kbReader.next();
        System.out.println("This is the next part of the String:");
    }
}
```
Multiple inputs:
In a similar way `nextInt()` and `nextDouble()` can be used multiple times to parse data input from the keyboard. For example, if 34 88 92 81 is input from the keyboard, then `nextInt()` can be applied four times to access these four integers separated by white space.

Inputting an entire line of text:
Inputting a `String` (it could contain spaces) from the keyboard using `nextLine()`:

```java
import java.io.*;
import java.util.*;
public class Tester {
    public static void main(String[] args) {
        Scanner kbReader = new Scanner(System.in);
        System.out.print("Enter your String here. "); //Enter One Two
        String s = kbReader.nextLine();
        System.out.println("This is my string,... " + s);
    }
}
```

Output would be as shown below:

```
Enter your String here. One Two
This is first part of the String, One
This is next part of the String, Two
```

Imports necessary:
We must import two classes, `java.io.*` and `java.util.*` that provide methods for inputting integers, `double`, and `String`. See Appendix I for more on the meaning of "importing".

Mysterious objects:
In the above three examples we used the following code:

```java
Scanner kbReader = new Scanner(System.in);
```

It simply creates the keyboard reader object (we arbitrarily named it `kbReader`) that provides access to the `nextInt()`, `nextDouble()`, `next()`, and `nextLine()` methods. For now just accept the necessity of all this...it will all be explained later.

The `Scanner` class used here to create our keyboard reader object only applies to 5.0xx or higher versions of Java. For older versions, see Appendix M for an alternate way to obtain keyboard input.

An anomaly:
Using a single `Scanner` object, the methods `nextInt()`, `nextDouble()`, `next()`, and `nextLine()` may be used in any sequence with the following exception:

It is not permissible to follow `nextInt()` or `nextDouble()` with `nextLine()`. If it is necessary to do this, then a new `Scanner` object must be reconstituted for use with `nextLine()` and any subsequent inputs.

Project... Going in Circles
The area of a circle is given by:

```
area = \pi r^2
```

Now, suppose we know the area and wish to find \( r \). Solving for \( r \) from this equation yields:

\[
\pi r^2 = \text{area} \\
\Rightarrow r = \sqrt{\frac{\text{area}}{\pi}}
\]

Write a program (project and class both named `RadiusOfCircle`) that uses `sqrt()` and \( \pi \) from the `Math` class to solve for the radius of a circle. Use keyboard input to specify the area (provide for the possibility of area being a decimal fraction).

Write out your solution by hand and then enter it into the computer and run. Before inputting the area, put a prompt on the screen like this.

```
What is the area? ...(the underscore indicates the cursor waiting for input)
```

Present your answer like this:

```
Radius of your circle is 139.4.
```

Project...What's My Name?
From the keyboard enter your first and then your last name, each with its own prompt. Store each in a separate `String` and then concatenate them together to show your full name. Call both the project and the class `FullName`. When your program is finished running, the output should appear similar to that below:

```
What is your first name? Cosmo
What is your last name? Kramer
Your full name is Cosmo Kramer.
```
Lesson 8. The boolean Type and boolean Operators

Back in Lesson 2 we looked at three fundamental variable types... int, double, and String. Here, we look at another very important type... boolean. This type has only two possible values: true or false.

Only two values:
Let's look at some statements that could come out either true or false. Suppose we know that \( x = 3 \) and also that \( y = 97 \). What could we say about the truth (or falseness) of the following statements?

1. \((x < 10) \text{ AND } (y = 97)\) Both parts are true so the whole thing is true.
2. \((x < 10) \text{ AND } (y = -3)\) First part is true, second part is false, whole thing is false.
3. \((x < 10) \text{ OR } (y = 97)\) If either part is true (both are) the whole thing is true.
4. \((x < 10) \text{ OR } (y = -3)\) If either part is true (first part is) the whole thing is true.

Correct syntax:
In the above examples there are three things we must change in order to have correct Java syntax:

1. System.out.println(true && false); //false
2. System.out.println(true && !false); //true
3. System.out.println(!b || c); //false
4. System.out.println((x > 102) && true); //false
5. System.out.println((jj == 1) || false); //false
6. System.out.println((jj == 40) && !false); //true
7. System.out.println(x != 3); //true
8. System.out.println(!!true); //true

Operator precedence:
Consider a problem like:

System.out.println((true && false) || (true && true) || false);

We can tell what parts we should do first because of the grouping by parenthesis. However, what if we had a different problem like this?

System.out.println(b && c); //false

Which part should we do first? The answers are different for the two different ways it could be done. There is a precedence (order) for the operators we are studying in this lesson (see Appendix H for a complete listing of operator precedence). The order is:

1. System.out.println( false && true || true); //true

We can tell what parts we should do first because of the grouping by parenthesis. However, what if we had a different problem like this?

System.out.println( (true && false) || (true && true) || false );

Example 1
System.out.println( true || false && false); //true
Do the false && false part first to get a result of false. Now do true || false to get a final result of true.

Negation (not) operator:
Another operator we need to know about is the not operator (!). It is officially called the negation operator. What does it mean if we say not true (true)? ... false, of course.

1. System.out.println(!true); //false
2. System.out.println(!false); //true
3. System.out.println(!(x < 5)); //false
4. System.out.println(!1 == 0); //true
5. System.out.println(!true); //true

Negation table:

<table>
<thead>
<tr>
<th>a</th>
<th>not a</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

Table 8-2 OR-ing

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
| true | true | true

Table 8-1 AND-ing

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>AND</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>
| true | true | true

Creation of booleans:
Create boolean variables as shown in the following two examples:

```java
boolean b = true;
boolean z = (p < 5) && (x + c);
```

Use the following code for example 1-10 below:
```java
int x = 79, y = 46, z = -3;
double d = 13.89, jj = 40.0;
boolean b = true, c = false;
```

1. System.out.println(true && false); //false
2. System.out.println(true && !false); //true
3. System.out.println(!b || c); //false
4. System.out.println((x > 102) && true); //false
5. System.out.println((jj == 1) || false); //false
6. System.out.println((jj == 40) && !false); //true
7. System.out.println(x != 3); //true
8. System.out.println(!!true); //true
9. System.out.println(!true); //false
10. System.out.println(b && c); //false

Operator precedence:
Consider a problem like:

System.out.println( (true && false) || (true && true) || false );

We can tell what parts we should do first because of the grouping by parenthesis. However, what if we had a different problem like this?

System.out.println( false && true || true); //true

Which part should we do first? The answers are different for the two different ways it could be done. There is a precedence (order) for the operators we are studying in this lesson (see Appendix H for a complete listing of operator precedence). The order is:

1. System.out.println( false && true || true); //true

We can tell what parts we should do first because of the grouping by parenthesis. However, what if we had a different problem like this?

System.out.println( (true && false) || (true && true) || false );

Example 1
System.out.println( true || false && false); //true
Do the false && false part first to get a result of false. Now do true || false to get a final result of true.
8.3

Example 2
System.out.println( true && false || false); //false
Do the true && false part first to get a result of false.
Now do false || false to get a final result of false.

Using a search engine:
You can use your knowledge of Booleans on the Internet. Go to your favorite search
engine and type in something like,

"Java script" and "Bill Gates"
and you will find only references that contain both these items.

On the other hand, enter something like,

"Java script" or "Bill Gates"
and you will be overwhelmed with the number of responses since you will get references
that contain either of these items.

You should be aware that the various search engines have their own rules for the syntax
of such Boolean searches.

8.4

Exercise for Lesson 8

In problems 1-5 assume the following:
int x = 23, y = -109;
double c = 2345.19, v = 157.03;
boolean a = false, s = true;

1. boolean gus = (x > 0) && (c == v);
   System.out.println(gus);

2. System.out.println(a || s);

3. System.out.println( (-1 * x) > 0 ) && !a);

4. boolean r = z = = x;
   System.out.println( r || false );

5. System.out.println( z!=x);

6. Fill in the following charts.
   \[
   \begin{array}{c|c|c}
   a & b & (a \&\& b) \\
   \hline
   false & false & false \\
   false & true & false \\
   true & false & false \\
   true & true & true \\
   \end{array}
   \]

7. Assume h, p, and q are booleans. Write code that will assign to b the result of AND-ing p and q.

8. Assign to the boolean variable w the result of OR-ing the following two things:
   A test to see if x is positive: A test to see if y equals z:

9. What are the two possible values of a boolean variable?

10. Write a test that will return a true if a is not equal to b. Assume a and b are integers.
    Store the result in boolean k. Do.

11. Write the answer to #10 another way.

12. What is the Java operator for boolean AND-ing?

13. What is the Java operator for boolean OR-ing?

14. System.out.println( (true && false) || ((true && true) || false) );

15. System.out.println(true && true || false);

16. System.out.println(true || true && false);

17. System.out.println(false || true && false);

18. System.out.println(false && true || false);

Now that we have learned to write a little code, it's time to turn to another part of our Computer
Science education. Computers haven't always been as they are today. Computers of just a few
years ago were primitive by today's standards. Would you guess that the computers that your
children will use someday would make our computers look primitive? Take a few minutes now
to review a short history of computers in Appendix S.
Lesson 9...•The if Statement

Now that we understand boolean quantities, let's put them to use in an if statement, one of Java's most useful "decision-making" commands. Consider the following code:

Example 1:

```java
//Get a grade from the keyboard
Scanner kbReader = new Scanner(System.in);
System.out.print(“What is your grade? ”);
int myGrade = kbReader.nextInt();

//Make a decision based on the value of the grade you entered
if(myGrade >= 70)
{
    //Execute code here if the test above is true
    System.out.println(“Congratulations, you passed.”);
}
else
{
    //Execute code here if the test above is false
    System.out.println(“Better luck next time.”);
}
```  

Leave off the else:

We do not necessarily always need the else part. Consider the following code without an else:

Example 2:

```java
Seamier kbReader = new Scanner(System.in);
System.out.print(“What state do you live in? ”);
String state = kbReader.nextLine(); //get state from keyboard
System.out.print(“What is the price? ”);
double purchasePrice = kbReader.nextDouble(); //get price from keyboard
double tax = 0;
if( state.equals(“Texas”))
{
    //Execute code here if test above is true
    tax = purchasePrice * .08; //8% tax
}
double totalPrice = purchasePrice + tax;
System.out.println(“The total price is “ + totalPrice + ”.”);
```  

It won't work!

There is just one difficulty with the above code in Example 2. It won't work! The problem is with how we are trying to compare two Strings. It cannot be as follows:

```java
        state.equals(“Texas”)
```

Rather, we must do it this way:

```java
        state.equalsIgnoreCase(“Texas”)
```

A good way to cover all the bases in the event someone mixes upper and lower case on the input is as follows:

```java
( state.equalsIgnoreCase(“Texas”) || state.equalsIgnoreCase(“Tx”) )
```  

What? No braces?

Braces are not needed if only one line of code is in the if or else parts. Likewise, the absence of braces implies only one line of code in if or else parts.

Example 3:

```java
int groovyDude = 37;
if(groovyDude = =37)
    groovyDude++; //this line is executed if test is true
System.out.println(groovyDude); //38
```  

Example 4:

```java
int groovyDude = 105;
if(groovyDude = =37)
    groovyDude++; //this line is not executed if test is false
System.out.println(groovyDude); //105
```  

The else if:

Multiple if's can be used in the same structure using else if.

Example 5:

```java
//Get a grade from the keyboard
Scanner kbReader = new Scanner(System.in);
System.out.print(“What is your grade? ”);
int theGrade = kbReader.nextInt();

if( theGrade>=90)
{
    System.out.println(“You made an A.”);
}
else if( theGrade>=80)
{
    System.out.println(“You made a B.”);
}
else if( theGrade>=70)
{
    System.out.println(“You made a C.”);
}
else if( theGrade>=60)
{
    System.out.println(“You made a D.”);
}
else
{
    System.out.println(“Sorry, you failed.”);
}
```
Exercise on Lesson 9

Use the following code for problems 1-10 and give the value of trueJWse for each:

```java
int i = 10, j = 3;
boolean true_false;
1. true_false = G > i;
2. true_false = (i > j);
3. true_false = (i >= j);
4. true_false = (j <= i) && G >= i);
5. true_false = (i > j) && G == 0);
6. true_false = (G < 50) || (G != 33);
7. true_false = (!j >= 0) || (i <= 50);
8. true_false = !(!(true));
9. true_false = (5 <= 5);
10. true_false = (j != i);

II. Write a statement that will store a true in boolean b if the value in the variable m is 44 or less.

12. Write a statement that will store a false in boolean b if the value in r is greater than 17.

13. What is returned by the following expression? (Recall that the precedence order of logical operators is !, &&, and finally ||)
   ![Expression](https://chart.example.com)
   In problem 14-16 what is the output?

14. String sl = "school BUS";
    if (sl.equals("school bus"))
        System.out.println("Equal");
    else
        System.out.println("Not equal");

15. String sl = "school BUS";
    if (sl.equalsIgnoreCase("school bus"))
        System.out.println("Equal");
    else
        System.out.println("Not equal");

16. intj = 19, m = 200;
    if (G == 18)
        m++;
    j++;
    System.out.println(m);
    System.out.println(j);

17. Write a statement that will store a false in boolean b if the value in g is not equal to 34.

18. Write a statement that will store a true in boolean b if integer k is even, false if it is odd.

19. Write a program that inputs a String from the keyboard after the prompt, "Enter your password". If it's entered exactly as "XR.ay", printout "Password entered successfully." otherwise, have it printout "Incorrect password."

20. What is output by the following "nested ifS" code?
    int k = 79;
    if (k < 50)
        
    
In problem 14-16 what is the output?

14. Enter an integer: 28
    The integer 28 is even.

15. Enter an integer: 2049
    The integer 2049 is odd.

16. Enter an integer: -236
    The integer -236 is even.
Quiz on Lesson 1

1. Create the "skeleton" of a program.

2. Write a line of code that will cause the word Hello to be printed.

3. Consider the following code:
   ```java
   System.out.print("Fire");
   System.out.println("Ants");
   ```
   Which of the following is actually printed?
   a. Fire Ants
   b. Fire
   c. Ant
   d. Fire

4. What is the syntax for indicating that a line of text is not Java code; rather, it isarem?
Quiz on Lesson 2

1. What are the three variable types we have studied up to this point?

2. Suppose you have the number 189.24. Which variable type would you use to store this number?

3. Write a line of code that declares k to be an integer.
   ```
   num, NIII, count, 12flag, flag-stuff, flagStuff; flag stuff
   ```

4. Which of the following are illegal names for a variable?
   num, NUIIL, count, 12fl, flag-stuff; flagStuff; flag stuff

5. Which of the following is the least desirable way to name a variable?
   redcolor, red_color, redColor

Key to Quiz on Lesson 2

1. What types we have studied up to this point?

2. Suppose you have the number 189.24. Which variable type would you use to store this number?

3. Write a line of code that declares k to be an integer.

4. Which of the following are illegal names for a variable?
   num, NIII, count, 12flag, flag-stuff; flagStuff; flag stuff

5. Which of the following is the least desirable way to name a variable?
   redcolor, red_color, redColor
Key to Exercise on Lesson 2

1. What are the three main types of variables used in JAVA and what are they used to store?
   a. String: stores names, letters, sentences, or any combination of characters, etc.
   b. int: stores integers, positive or negative
   c. double: stores decimal fraction numbers

2. What type of variable would you use to store your name?
   String

3. What type of variable would you use to store the square root of 2?
   double: square root of 2 is approximately 1.414

4. What type of variable would you use to store your age?
   int

5. Write a single line of code that will create a double precision variable called \( p \) and store \( 1.921 \times 10^{-16} \) in it.
   ```java
double p = 1.921E-16;
```

6. Write a single line of code that will create an integer variable called \( i \) and store 407 in it.
   ```java
   int i = 407;
   ```

7. Write a single line of code that will create a String variable called \( my\_name \) and store your name in it.
   ```java
   String my\_name = "Barney Fife";
   ```

8. Write a line of code that will declare the variable \( count \) to be of type int. Don't initialize.
   ```java
   int count;
   ```

9. Write a line of code that initializes the double precision variable \( bank\_Balance \) to \( 136.05 \).
   ```java
   Assume this variable has already been declared.
   double bank\_Balance = 136.05;
   ```

10. Which of the following are legal variable names?
    scooter13   19\_scooter   homer-5   _mary   public   doubled   double   ab c

11. Which of the following is the most acceptable way of naming a variable. Multiple answers are possible.
    a. GroovyDude
    b. GROOVYDUDE
    c. groovyDude
    d. GroovyDude
    e. groovy\_dude
    f. groovydude

12. Comment on the legality of the following two lines of code.
    ```java
    double dist = 1003; // legal
    int alt = 1493.86; // illegal
    int num = 1_2_3; // legal
    ```

Quiz on Lesson 3

1. What is output by the following code?
   ```java
   Strings = "Mona Lisa";
   System.out.println(s.length());
   ```

2. What is output by the following code?
   ```java
   String girl = "Heather Jones";
   System.out.println(girl.substring(8));
   ```

3. What is output by the following code?
   ```java
   String s = "Beaver Cleaver";
   System.out.println(s.toUpperCase());
   ```

4. What is the index of the 'L' in the String "Abraham Lincoln"?

5. What is output by the following code?
   ```java
   String s = "Beaver Cleaver";
   System.out.println(s.toUpperCase());
   ```
9. Write code that will produce the following printout using only a single `println()`:
   ```java
   Hello
   Hello again
   System.out.println("Hello and Hello again");
   ```

10. Write code that will produce the following printout.
    A backslash looks like this \\, ...right?
    ```java
    System.out.println("A backslash looks like this \\
    \, \*right\?");
    ```

11. What is output by the following?
    ```java
    String pq = "Eddie Haskell";
    int hm = pq.length();
    String ed = pq.substring(hm - 4);
    System.out.println(ed);
    ```

12. Which character is at the 5th index in the String "Herman Munster".
    ```java
    ...
Quiz on Lesson 4

1. If int j = 3, what will be the value of j++?

2. What is another way to write \( p \leftarrow p - 1 \)?

3. Write \( x \leftarrow x + j \); another way.

4. Write code that is equivalent to saying the new value of \( w \) is the old value of \( w \) plus 6.

5. Which of the following is illegal?
   a. \( x = 9 \);
   b. \( 9 = x \);

Key to Quiz on Lesson 4

1. If int j = 3, what will be the value of j++?

2. What is another way to write \( p \leftarrow p - 1 \)?

3. Write \( x \leftarrow x + j \); another way.

4. Write code that is equivalent to saying the new value of \( w \) is the old value of \( w \) plus 6.

5. Which of the following is illegal?
   a. \( x = 9 \);
   b. \( 9 = x \);
Key to Exercise on Lesson 4

Unless otherwise directed in the following problems, state what is printed. Some of these problems may have incorrect syntax and in those cases you should answer that the code would not compile:

1. int h = 103;
   int p = 5;
   System.out.println(++h + p); //109
   System.out.println(h); //104

2. Give three code examples of how to increment the integer } by 1.
   j = j + 1;
   j++; // for ++j;
   j+=1;

3. double def;
   double f = 199237;
   def = f;
   System.out.println(def); //199237

4. Write a single line of code that will print the integer variable zulu and then decrement its value by L
   System.out.println(zulu-);

5. int a = 100;
   int b = 200;
   b=a;
   System.out.println(b + 1); //3

6. Write a single line of code that does the same thing as #6 but without using -=
   v = v - (p-30); // -22

7. Write a single line of code that does the same thing as #6 but without using -=
   v = v - (p-30); // -22

8. int p = 40;
   int q = 4;
   System.out.println(2 + 8 * q/2 - p); // -22

9. int sd = 12;
   int x = 4;
   System.out.println(sd0x); //2
   System.out.println(x); // 5

10. int g;
    3 = g;
    System.out.println(g*79); //3
    What is the result? Won't compile •••3 = g16 illegal

11. On a single line of code declare m, b, and to be double and on that same line initialize them all to be 3.14.
    double m = 3.14, b = 3.14, f = 3.14;

12. On a single line of code declare x, y, and z all to be of integer type.
    int x, y, z;

13. int m = 36;
    int j = 5;
    m = m/j; // new m is old m divided by j
    System.out.println(m);
    What's printed? 7

14. System.out.println(3/4 + 5*2/33 -3 +8*3);
    What's printed? 21

15. What is the assignment operator?

16. Write a statement that stores the remainder of dividing the variable } by } in a variable named k
    k = i%j;

17. int j = 2;
    System.out.println(7%3 + j++ + G -2j); //4

18. Show three different ways to decrement the variable j.
   j--; -j; j = j-1;
Project... Cheating on Your Arithmetic Assignment

```java
public static void main(String[] args)
{
    int answ = 7 * (182 - 68) - 7 + 19;
    System.out.printf("%d \n", answ);
    System.out.println("\n" + answ);
}
```

1. Which of the following is illegal?
   a. `double d = 27;`
   b. `int i = 203.932;`

2. The following code is illegal. Rewrite the code using an integer cast to make it legal.
   ```java
double d = 187.2;
int j = d;
```

3. What is the output of the following?
   ```java
   System.out.println(27/5 + 3.1);
   System.out.println((double)7/2 + 3.1);
   ```

Quiz on Lesson 5

1. Which of the following is illegal?
   a. `double d = 27;`
   b. `int i = 203.932;`

2. The following code is illegal. Rewrite the code using an integer cast to make it illegal.
   ```java
double d = 187.2;
int j = d;
```

3. What is the output of the following?
   ```java
   System.out.println(27/5 + 3.1);
   System.out.println((double)7/2 + 3.1);
   ```
Key to Exercise on Lesson 5

Unless otherwise instructed in the following problems, state what gets printed.

1. Write code that will create a constant E that's equal to 2.718.
   final double E = 2.718;

2. Write the simplest type constant that sets the number of students, NUM STUDENTS, to 236.
   final int NUM_STUDENTS = 236;

3. What's wrong, if anything, with the following code in the main method?
   final double Area; //Nothing is wrong
   Area = 203.49;

4. int cnt=27; //Won't even compile. This line is illegal
   System.out.println(cnt);

5. double d = 78.1;
   int fg = (int)d;
   System.out.println(fg); //78

6. Is double f4 = 22; legal? Yes

7. The following code stores a 20 in the variable j:
   double j = 61/3; //What small change can you make to this single line of code to make it produce the "real" answer to the division?
   double j = (double)61/3; or double j = 61.0/3;

8. System.out.println( (double)(90/9) ); //10.0

9. System.out.println( 4 + 6.0/4 + 5 *3-3); //17.5

10. int p = 3;
    double d = 10.3;
    int j = (int)d;
    System.out.println(p + p * d -3 * j); 1118.9

11. int p = 3;
    double d = 10.3;
    int j = (int)d;
    System.out.println(p + p * (int)d -3 * j); 1118

The following code applies to 12-15:

    int dividend = 12, divisor = 4, quotient = 0, remainder = 0;
    int dividend2 = 13, divisor2 = 3, quotient2 = 0, remainder2 = 0;
    quotient = dividend/divisor;
    quotient2 = dividend2/divisor2;
    remainder = dividend % divisor;
    remainder2 = dividend2 % divisor2;

12. System.out.println(quotient); //3
13. System.out.println(remainder); //0
14. System.out.println(quotient2); //4
15. System.out.println(remainder2); //1

16. Write a line of code in which you divide the double precision number d by an integer variable called i. Type cast the double so that strictly integer division is done. Store the result in an integer.
   int j = (int)d / i;

17. Suppose we have a line of code that says:
    final String M = "ugg";

    Later in the same program, would it be permissible to say the following?
    M = "woW";

    No, a constant can't be changed.

18. Is the following code legal? If so, what is printed? If not, why?
    int k = 7;
    k *= 5;
    System.out.println(k);

    Yes, legal... k * 5 is printed. k *= 5 is equivalent to k = (int)(k * 5);
Quiz on Lesson 6

1. Show how we would calculate and print the square root of 139.46.

2. Show how we would print the value of \( n \).

3. What would the following print if \( d = 208.4 \)?
   ```java
   System.out.println(Math.ceil(d));
   ```

4. Write a line of code that will calculate \((3.1)^{0.02}\) and store the result in double \( d \).

Key to Quiz on Lesson 6

1. Show how we will calculate and print the square root of 139.46.

2. Show how we would print the value of \( n \).

3. What would the following print if \( d = 208.4 \)?
   ```java
   System.out.println(Math.ceil(d));
   ```

4. Write a line of code that will calculate \((3.1)^{0.02}\) and store the result in double \( d \).
Key to Exercise on Lesson 6

1. Write code that will take the square root of \( x \) and store the result in \( y \).
   ```java
double y = Math.sqrt(x);
```

2. Write code that will multiply the value of the integer \( j \) times the absolute value of the integer \( m \) and then store the result in the integer \( k \).
   ```java
   int k = j * Math.abs(m);
   ```

3. Is the following legal? If not, what would you do to make it legal?
   ```java
   int k = Math.abs(127.5);
   ```
   No, it tries to store a double in an int. There are two things you could do:
   ```java
   double k = Math.abs(-127.5);
   ```
   ```java
   int k = (int)Math.abs(-127.5); //this is the worst because you would lose some decimal places
   ```

4. Write a statement that will print the result of \( 2^{1.5} \).
   ```java
   System.out.println(Math.pow(2, 1.5));
   ```

5. System.out.println(Math.ceil(-157.2)); // -157.0

6. System.out.println(Math.floor(-157.2)); // -158.0

7. System.out.println(Math.ceil(157.2)); // 157.0

8. System.out.println(Math.floor(157.2)); // 157.0


10. System.out.println(Math.ceil(157.7)); // 158.0

11. System.out.println(Math.ceil(157)); // 157.0

12. System.out.println(Math.ceil(157.7)); // 158.0

13. Write a statement that will print the natural log of 18 ... same as \( \ln(18) \) on a calculator.
    ```java
    System.out.println(Math.log(18));
    ```

14. Write a line of code that multiplies \( \text{double } p \) times \( t \) and stores the result in \( b \).
    ```java
    double b = p * Math.PI;
    ```
Quiz on Lesson 7

1. What are the three data types that we are able to input from the keyboard?

2. Suppose a `Scanner` object, `kbReader`, has already been created. Show code that uses `kbReader` to input a number with "decimal places" from the keyboard and store the result in the variable, `fract`.

3. Suppose a `Scanner` object, `kbReader`, has already been created. Show code that uses `kbReader` to input a quantity like "Jo mamma" from the keyboard and store the result in the variable, `name`.

4. Suppose a `Scanner` object, `kbReader`, has already been created. Show code that uses `kbReader` to input a number with "no decimal places" from the keyboard and store the result in the variable, `count`.

Key to Quiz on Lesson 7

1. What are the three data types that we are able to input from the keyboard?

2. Suppose a `Scanner` object, `kbReader`, has already been created. Show code that uses `kbReader` to input a number with "decimal places" from the keyboard and store the result in the variable, `fract`.

3. Suppose a `Scanner` object, `kbReader`, has already been created. Show code that uses `kbReader` to input a quantity like "Jo mamma" from the keyboard and store the result in the variable, `name`.

4. Suppose a `Scanner` object, `kbReader`, has already been created. Show code that uses `kbReader` to input a number with "no decimal places" from the keyboard and store the result in the variable, `count`. 
Quiz on Lesson 8

1. What are the two possible values for a boolean type variable?

2. If boolean p is false what is !p?

3. What is the operator used to indicate Boolean AND?

4. What is the operator used to indicate Boolean OR?

5. Which has higher precedence && or ||?

   System.out.println(bb);

6. What is the output of the following code assuming that p is true and q is false?
   
   boolean bb = !p || q;

Key to Quiz on Lesson 8

1. What are the two possible values for a boolean type variable?

2. If boolean p is false what is !p?

3. What is the operator used to indicate Boolean AND?

4. What is the operator used to indicate Boolean OR?

5. Which has higher precedence && or ||?

   System.out.println(bb);

6. What is the output of the following code assuming that p is true and q is false?
   
   boolean bb = !p || q;
Key to Exercise for Lesson 8

In problems 1-5 assume the following:
int z = 23, x = -109;
double c = 2345.19, v = 157.03;
boolean \(a\) = false, \(s\) = true;

1. boolean \(gus\) = \((x > 0) \&\& (c = = v)\); System.out.println(\(!gus\)); //true

2. System.out.println(\(a \mid \mid s\)); //true

3. System.out.println(\((\ (-1 * x) > 0) \&\& !a\)); //true

4. boolean \(r\) = \(z = = ;\); System.out.println(\(r \mid \mid false\)); //false

5. System.out.println(\(z = = x\)); //true

6. Fill in the following charts

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>((a &amp;&amp; b))</th>
<th>(a)</th>
<th>(b)</th>
<th>((a \mid \mid b))</th>
</tr>
</thead>
<tbody>
<tr>
<td>ful-O</td>
<td>ful-O</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>truo</td>
<td>false</td>
<td>truo</td>
<td>truo</td>
<td>truo</td>
</tr>
</tbody>
</table>

7. Assume \(p, q\), and \(q\) are booleans. Write code that will assign to \(b\) the result of AND-ing \(p\) and \(q\).

\(b = p \&\& q;\)

8. Assume to the boolean variable \(w\) the result of OR-ing the following two things:

A test to see if \(x\) is positive: A test to see if \(y\) equals \(z\):

\(w = (x > 0) \mid \mid (y = = z);\)

9. What are the two possible values of a boolean variable?

true, false

10. Write a test that will return a true if \(x\) is not equal to \(b\). Assume \(a\) and \(b\) are integers.

Store the result in boolean \(kDog\):

boolean \(kDog\) = \(a = = b;\)

11. Write the answer to #10 another way.

boolean \(kDog\) = \(!a = = b;\)

12. What is the Java operator for boolean AND-ing? \&\&

13. What is the Java operator for boolean OR-ing? \mid \mid

14. System.out.println(\((true \&\& false) \mid \mid (true \&\& false)\)); //true
Quiz on Lesson 9

1. Show the basic skeleton of an if-else structure.

2. In the following code, assume that the portion designated with <#1> is a true statement. What will be the output?
   
   ```java
   if (<#1>)
   
   System.out.println("Elvis");
   
   System.out.println("Presley");
   ```

3. In problem 2, what would be the output if <#1> was a false statement?

4. If you wanted to compare the contents of two Strings, s1 and s2, which of the following would be appropriate to use?
   
   a. `s1.equals(s2)`
   b. `s2.equals(s1)`
   c. `s1.equalsIgnoreCase(s2)`
   d. `All of the above`
   e. `s1 == s2`
   f. `None of these`

Key to Quiz on Lesson 9

1. Show the basic skeleton of an if-else structure.

2. In the following code, assume that the portion designated with <#1> is a true statement. What will be the output?
   
   ```java
   if (<#1>)
   
   System.out.println("Elvis");
   
   System.out.println("Presley");
   ```

3. In problem 2, what would be the output if <#1> was a false statement?

4. If you wanted to compare the contents of two Strings, s1 and s2, which of the following would be appropriate to use?
   
   a. `s1.equals(s2)`
   b. `s2.equals(s1)`
   c. `s1.equalsIgnoreCase(s2)`
   d. `All of the above`
   e. `s1 == s2`
   f. `None of these`
Key to Exercise on Lesson 9

Use the following code for problems 1 - 10 and give the value of true/false for each:
```
int i = 10, j = 3;
boolean true_false;
```
1. true_false = G > i; //false
2. true_false = (i > j); //true
3. true_false = (i==j); //false
4. true_false = (G <= i) || G > i); //true
5. true_false = ( (i>j) && G == O); //false
6. true_false = (G < 50) || G != 33); //true
7. true_false = ( G >= 0) || (i <= 50); //true
8. true_false = ( !(true)); //false
9. true_false = (5 < 5); //true
10. true_false = G != i); //true

11. Write a statement that will store a true if the value in variable m is 44 or less.
    boolean b = m <= 44;
12. Write a statement that will store a false if the value in r is greater than 17.
    boolean b = !(r > 17); or boolean b = (r <= 17);
13. What is returned by the following expression? (Recall that the precedence order of
    logical operators is !, &&, and finally ||.)
    !(2>3) || GS && (7>1) && (4<15) || (35<36) && (89>34)
    false

In problems 14 – 16 state what's output.

14. String sl = "school BUS";
    if(sl.equals("school bus")
    System.out.println("Equal");
    else
    System.out.println("Not equal");
15. String S1 = "school BUS";
    if(sl.equalsIgnoreCase("school bus")
    System.out.println("Equal");
    else
    System.out.println("Not equal");
16. int j = 19, m = 200;
    if(G == l&k)
        m++;
    else
        System.out.println(m); //200
    System.out.println(j); //20

17. Write a statement that will store a false if the value in g is not equal to 34.
    boolean b = g = 34;
18. Write a statement that will store a true if integer k is even, false if it is odd.
    boolean b = (k%2) == 0;
19. Write code for the main method that inputs a String from the keyboard after the prompt,
    "Enter your password". If it's entered exactly as "XRay", printout "Password entered
    successfully."; otherwise, have it printout "Incorrect password.".
    Scanner kbReader = new Scanner(System.in);
    System.out.println("Enter your password.");
    String passWord = kbReader.nextLine();
    if(passWord.equals("XRay")
    System.out.println("Password entered successfully.");
```
Test Through LCssO:ri-3

1. What is the meaning of a "floating point" number?

2. Write the single line of code that will cause Computer science to be printed on the screen.

3. Write two lines of code where the first prints "Hello", the second prints "world", and the net result is that they print together exactly as follows:
   Hello world

4. Suppose you want to document your program with the following text
   Program author: Miss Fefe LaFu
   Modify this line of text so that it can legally reside in your program as a comment.

5. What variable type would you use to store your school's name?

6. What's wrong (if anything) with this line of code?
   int ii = 59;

7. What's wrong (if anything) with this line of code?
   double dur = 102;

8. What's wrong (if anything) with this line of code?
   int pugh = 1003.84;

9. What does rem stand for?

10. The following block of text is to be considered a remark. Rather than putting a double slash in front of each line of text, modify the text using the "block" technique of "commenting-out" the text.
    Name: Elvis Presley
    Date: Jan 34, 1776
    School: Nashville High School

11. Write the following number in a more conventional way as it would be done in a math class.

12. In the following two lines of code, state which line is initializing and which is declaring.

13. Suppose we wish to create a variable name using the words distance to line. Which of the following is the conventionally acceptable way to name this variable? (There may be more than one answer.)

14. What is output by the following code?
   String s = "My dog";
   int theLen = s.length();
   System.out.println(theLen);

15. What is output by the following code?
   String gg = "Wiggle worm";
   String s = gg.substring(3);
   System.out.println(s);

16. What is output by the following code?
   String s = "My dog";
   int theLen = s.length();
   System.out.println(theLen);

17. What is output by the following code?
   String gg = "Wiggle worm";
   String s = gg.substring(3);
   System.out.println(s);
22. What is output by the following code?
   ```java
   String gg = "Big bad Wolf";
   String s = gg.substring(2, 6);
   System.out.println(s);
   ```

23. Which character is at the 6th index in "Hello there"?

24. What is output by the following code?
   ```java
   String xc = "Big Wally";
   System.out.println(xc.toLowerCase());
   ```

25. Suppose that you already have a String called dep. Write two lines of code, the first of which will assign to the String df the String dep converted to all capital letters. The second line should print df

26. Write a line of code that will print \textbf{We have "bad" enough.}

27. What is output by the following code?
   ```java
   String pgr = "ijklm
   opqrs";
   System.out.println(pgr);
   ```

28. What is the escape sequence for a backslash?

29. Write code that will determine the number of characters in the String \(v\) and then print, \textbf{The String has 22 characters.} The 22 part is just an example. Use the \textit{length()} method to print the 22 (or whatever it might be actually be for \(v\)).

30. What will be the value of \(jy\)?
    ```java
    String bj = "Oh my goodness!";
    int pl = 4;
    String jy = bj.substring(0, pl);
    ```

31. What will be the output of the following code?
    ```java
    String s = "Kitty cat";
    int kc_len = s.length();
    String ms = s.substring(kc_len-2);
    System.out.println(ms);
    ```

32. What is the meaning of "concatenatin,..., furo-"frh. gi \textbf{tb} ilie?"?

33. Give the names of two methods of the \textit{String} class.
Test Through Lesson 7 (Emphasizing 4-7)

1. Write a single line of code to accomplish the following:
   Declare p, q, and r to be of type int. Initialize p to a value of 3 and r to a value of -16.

2. Which of the following is legal assuming w has been declared as an int type.
   a. -59 = w;
   b. w = -59;
   c. Both a and b
   d. None of these

3. What is output by the following code?
   int x = 22;
   int y = 6;
   System.out.println(x % y);

4. Write the "skeleton" code for a class called Bozo. This includes the skeleton of the class plus the skeleton of the main method inside it.

5. Consider the following code fragment:

```java
Scanner kbReader = new Scanner(System.in);
System.out.println("What is your code? ");
<#1>
// this line of code stores in m whatever is typed from the keyboard in response to the above prompt
System.out.println("Your code is " + m + ");
```

Now suppose when this code is nm and the program finishes, the output screen appears as follows:
What is your code? 1234
Your code is 1234.

To achieve all this, what would be a legal replacement for #1 above? (Assume that the data to be input is always just digits as in the example.)
   a. String m = kbReader.nextLine();
   b. int m = kbReader.nextInt();
   c. double m = kbReader.nextDouble();
   d. All of the above
   e. None of these