AP Computer Science A

Summer assignment

Hello, welcome to AP Computer Science A. This should be a fun class with lots of learning to do. To kick start your 2018-19 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

Blue Pelican Java

by Charles E. Cook

Version 7.0.lA

"Blue Pelican Java," by Charles E. Cook. ISBN 1-58939-758-4.

Preface

You will fmd 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 _defi.nitions 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. Ifyou 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, antl 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 Progranuning) 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. Inthose 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 "robusf' inthat there is not great depth for some of the topics. Actually the depth is there,... inthe 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 inthe lessons. As an example, in the *split* method is introduced. The *split* method uses regular expressions that are briefly discussed there; however, the in-depth presen1ation of regular expresSions is placed in Appendix AC.

Unfortunately, this book does not introduce any graphics or windows prOiramming. The 57 lessons inthis book can be covered in one school year, but just barely. To prepare students for the AP test (and contests) there is onJy time to cover the essentials presented in this book. Check httpJ/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.

Jn this book you will also notice another part of my philosophy of teacJ:iing and educational material ingenera!...Keep it simple... I try to keep things as simple and uncluttered as possible. For example, you will find specific examples ingreater 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 fonnat 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 "contesttypc" 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. Hopefully, I 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 Bluel and it was the first my students used. I always thought BlueJ was an elegant name and had expressed a desire to a collėlle 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 ungainly, clunky bird. For the lack of an existing name for the book during development, it continued to be called Blue Pelican. Ifyou 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

Table of Contents

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".

"1

Golden Nuggets of Wisdom arc short learning/review activities. In the six weeks preceding an AP exam, contest, or other major evaluation. study one of these each day. Follow up with a quiz (provided in the Teacher's Test/Answer Book) on that topic the next day.

H [- '

Lesson 1.....Hello World

Prograln Skeleton:

Enter the following program skeleton, compile (prep'.11°e it to nµi), and .then run.(execute). Your instructor may have you give it a specific project naine; o'iherW:ise;' call the project *Lesson].*

If you do not know how to enter and execute a program, aSk yOU:r inStriictor, or use the appendices in this book for two of the more popular program.ming environments. See Anpendix N for the BlueJ environment and A0ne'.nilix o for the JCreator environment.

```
public class Tester
{
        public static void main(String argsO)
        {
```

At this point don't worry about what any of this means. It's just sometlllng 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 woi:d, method. We will constantly refer to methods throughout this course.) We will also add a remark.

```
public class Tester //We could put any name here besides Tester
{
        public static void main(String argsQ)
        {
                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 printout 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 / 7.0445902
5. ·13·* 159S6

Two *printlns* for the price of one:

Next, modify your program so that the *main* method looks as follows:

```
public static void main(String args0)
{
        System.outprintln("Hello world");
        System.outprintln("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:

```
public static void main(String argsO)
{
        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 *conceming 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:

```
public class Tester
{
        //Programmer: Kosmo Kramer
        //Date created: Sept 34, 1492
        //School: Charles Manson High School; Berkley, Ca

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

Block rems:

It can get a little tedious putting the double slaSh rem-indkator infront of each . line, especially if we have quite a few remark lines. Inthis case we can "block · rem" all the comment lines as follows:

```
public class Tester
{
        /*Programmer: Kosmo Kramer
        Date created: Sept 34, 1492
        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.

Project...From Me To You

Create a new project called *FromMeToYou* having a *Tester* class with the following content Also include remarks above *public class Tester* that identifies you as the author along with the date of creation of this program:

```
//Author: Charles Cook
//Date created: Mar 22, 2005
public class Tester
{
   public static void main(String argsO)
   {
```

Supply code in the place of ... that will produce the following printout

```
From: Bill Smith
Address: Dell Computer, Bldg 13
Date: April 12, 2005

To: Jack Jones

Message: Help! I'm trapped inside a computer!
```

Lesson 2.....Variable Types *(String, int, doubie)*

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.)

I. *String*....used to store things in quotes....like "Hello world"

Sample code:

```
public static void main(String argsO)
{
        String s = "Hello cruel world";
        System.out.println(s);
```

2. *int* ....used to store int.egers (positive or negative)

Sample code:

```
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,

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:

```
public static void main(String args[])
{
        double d = -137.8036;
        System.out.println(d);

        d = l.45667E23: .//Scientific notation. .means l.45667 X 10^23
```

Declaring and initializing:

When we say something like

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 "fif' 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 pennissible inside the nrupe is the underscore (" "). Variable names cannot be one of the reserved words (key words...see Appendix A)that are part of the Java language.

Legal names	Illegal names
Agro	139
D	139Abc
d31	fut Ono
hoppergee	class
hopper_gee	slow.Sally
largeArea	double
go!dNugget	gold;Nugget
	hopper-gee

Variable naming conventions:

It is traditional (although not a hard and fust 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 iand 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?
    scooterl3   139_scooter   homer-5   ;mary   public   doubled   double   ab c

I L Which of the tOllowing 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:

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

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

```
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:

```
System.outprintln("Hello"  +" good buddy");  // prints Hello good buddy
```

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

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

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

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

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

```
S p a r k y  t h e   d o g .
0 1 2 3 4 5 6 7 8 9 10 11 12 13
```

so now we see that the 'k' has index 4 and we go from k all the way to the end of the string to get "lcy the dog".

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

```
String myPet = "Sparky the dog";
String smallPart = myPetsubstring(4,  12);
System.outprintln(smallPart);   //prints ky the d
```

How do we get ky *the ff!* 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.*

Conversion between lower and upper case:
toLowerCase converts all characters to lower case (small letters)

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

toUpperCase converts all characters to upper case (capital letters)

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

Note: *length, substring, toLowerCase,* and *toUpperCase* 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:

```
int x = 27;
String s = "Was haben wir gemacht?";  //German for "What have we done?"
String combo = s + " " + x;
System.outprintln(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):

```
What "is" the right way?
```

```
String s = "What \"is\" the right way?";
System.out.println(s);  //prints What  "is" the right way?
```

Another escape sequence, \n, will create a new line (also called line break) as shown below:

```
String s = "Here is one line\nand here is another.";
System.outprintln( s);
```

Prints the following:

```
Here is one line
and here is another.
```

The escape sequence, \\, will allow us to print a backslash within our *String.* Otherwise, if we try to insert just a single \ it will *be* interpreted as the beginning of an escape sequence.

```
System.out.println("Path = c:\\nerd_file.doc");
```

Prints the following:

```
Path    c:\nerd_file.doc
```

The escape sequence, \t, will allow us to "tab" over. The following code tabs twice.

```
System.out.println("Nrune:\t\tAddress:  ");
```

Prints the following:

```
Name:               Address:
```

Exercise on Lesson 3

L 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 *Systemout.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 *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 follov.'ing?

```
All "good" men should come to the aid of their country.
```

9. Write code that will produce the following prtOut Using.onIY a in£-Je *priiltln().*

```
Hello
Hello again
```

10. Write code that will produce the following printout

```
A backslash looks like this \, ...tight?
```

11. What is output by the following?

```
String pq = "Eddie Haskel";
int hm = pq.length( );
String ed = pq.substring(bm - 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:

```
Allan Alda>>>lan A
John Wayne>>>hn Wa
Gregory Peck>>>egory P
```

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.

```
int i = 3;  II Ok,...assign the value 3 toi Notice the direction of data flow.
    VJ
3      ; JI Illegal! Data never flows this way!
```

```
double p;
double j = 47.2;
p = j;  JI assign the value of j to p. Both p and j are now equal to 472
```

Multiple declarations:

It is possible to declare several variables on one line:

```
double d, mud, puma;  //the variables are only declared
double x = 31.2, m = 37.09, zu, p = 43.917;  //x, m, & p declared and initialized
                                             II zu is just declared
```

Fundamental arithmetic operations:

The basic arithmetic operation are +,-,* (multiplication), I (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"lob* 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.)

```
System.out.println(5 + 3 * 4 -7); //10
System.out.println(S -5*6 I 3 + (5-6) * 3); 11-5
```

Not the same as in Algebra:

An unusual assignment. ...consider the following:

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

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:

```
x++;  means the same as        x = x +l;
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)

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

Componnd operators:

	Sxntax Examp:le		Simolified meanine:
a.	x += 3;	7	x = x + 3;
b.	x -= y - 2;	7	x = x-(y - 2);
'. ■	z*= 46;	7	z = z * 46;
d. /=	p/= x-z;	7	p = p I (x-z);
'. %=	jo/o=2	7	j = jo/o2;

Code Examples

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

double d = 20.3;
double m =IO.O;
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*.

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.

<u>Code Examples</u>

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

int q= 78;
int p = ++q + 2;
System.out.println("p =" + p +",q=" + q); //p = 81, q = 79
```

Integer division truncation:
When clividing two integers, the fractional part is truncated (thrown away) as illustrated by the following:

```
int x = 5;
int y = Z;
System.out.println(x J 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.outprintln(++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(det);
   ```

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

5.
   ```
   int a = lOO;
   int b = 200;
   b/=a;
   System.oulprintln(b +1);
   ```

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?

1L On a single line of code declare *m, h,* andfto 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 zall 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. intj= 2;
System.outprintln(7%3 +j++ + G -2));

18. Show three different ways to decrement the variable *J-*

Project..• Cheating on Your Arithmetic Assignment

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

79 +3 * (4 + 82 -68) -7 +19

(179 +21+10) / 7 + 181

10389 * 56 * 11 + 2246

The printout should look like the following:

```
79 + 3 * (4 + 82 - 68) - 7 + 19    145

(179 + 21 + 10) / 7 + 181 = 211

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

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 Jose 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:

I. An example of when we would lose information:

```
double 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 We can force compilation and therefore result in 29.78 being "stored" in *i* as follows (actually,just 29 is stored since *i* can only hold integers):

int i = (int)d; //(int) "casts" *d as* an integer. . It converts dto integer form.

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

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

The most pl'"ecise:

In a math operation involving two different data types, the result is given in terms of the mol'"e precise of those two types...as in the following example:

```
int i = 4;
double d = 3;
double ans = i/d; //ans will be 1.33333333333333...the result is double precision
```

20 +*5* * 6.0 returns a *double.* The 6.0 might look like an integer to us, but because it's written with a decimal point, it is considered to be a floating point number...a *double.*

Some challenging examples:

What does 3 + 5.0/2 + 5 * 2 - 3 return? 12.5

What does 3.0 + *512* + 5 * 2 - 3 return? 12.0

What does (int)(3.0 + 4)/(1 + 4.0) * 2 - 3 return? -.2

Don't be fooled:

Consider the following two examples that are very similar...but have different answers:

```
double d = (double)5/4;  //same as 5.0 / 4  ..(double) only applies to the 5
System.oulprintln{d);  //1.25

intj =5;
int k = 4;
double d = (double)G I k); JIG / k) is in its own little "world" and performs
                                   //integer division yielding 1 which is then cast as
                                   //a double, 1.0
System.out.println(d);  //I.0
```

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 al! capital letters. The rules for legal constant names are the same as for variable names. Following is an example of a constant:

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

The following illustrates that constants can't be changed:

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

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

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

Constants can also be of type *String, inf* and other types.

```
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 j+ = *x;* was equivalent *to j =j + x;*. Actually, for all compound operators there is also an implied cast to the type *of).* For example, *ifj* is of type *int,* the real meaning of *j+= x;* is:

```
j = (int)G + 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 O for the parenthesis.

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

```
Problem I: 572 * (il / i2) +1
Problem 2: 572 * ( (double)il I i2 ) + I
Problem 3: 15-il * ( dl * 3) + 4
Problem 4: 15-il * (int)( dl * 3) + 4
Problem 5: 15-i1* ((int)dl * 3) + 4
```

Your printout should look like the following:

```
Problem 1: LO
Problem 2: 39.13333333333333
Problem 3: -1345.39999999999
Problem 4: -1337
Problem 5: -1313
```

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.

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

3. What's wrong, if anything, with the following code in the *main* method?
   ```
   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 *doublef4 = 22;* legal?

7. The following code stores a 20 in the variable):
   ```
   doublej = 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));

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

10. int p = 3;
    double d = I0.3;
    int j = (int)5.9;
    System.out.println(p + p * d -3 * j);

11. int p = 3;
    double d = 103;
    intj = (int)5.9;
    System.out.println(p + p * (lnt)d -3 *j);

The following code applies to 12   15:

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

12. System.out.println(quotient);

13. System..out.println(remainder);

14. System.out.println(quotient2);

15. System.out.println(remainder2);

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 inj, an integer.

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 = '\•ow";

18. Is the following eode legaJ? If so, what is printed? If not, why?

```
int k = 7;
k*=.5;
System.out.println(k);
```

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 l7 and store the result inp, do the following:

```
double p = Math.sqrt(17);
```

Notice that we must store the result in a *double* ...p in this case. We must store in a *douhle* 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 method 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 !he 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
"rt	double sqrt(double x)	Returns the square root of x
ceil	double ceil(double x)	Returns next highest whole number from x
floor	double floor(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.)

Method	Signature	Description
random	double random( )	Returns a random double (range O:S r < 1)
round	long round(double x)	Returns x rounded to nearest whole number
PI	double PI	Retums 3.14159625..

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

L double d = -379.22;
   System.out.println(Math.abs(d)); //379.22

2. double b = 4201;
   double e = 3.728;
   System.outprintln (Math.pow(b, e)); //1126831.027

3. double d = 2034.56;
   System.out.println(Math.sqrt(d)); //45.10609715

4. double d = 1.4;
   Systemout.println(Math.ceil(d)); //2.0

5. double d = -L6;
System.outprintln{ Math.ceil(d)); //-1.0

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

7. double d = -1.6;
System.outprintln(Matb.floor(d)); //-2.0

The last four examples illustrating *floor* and *ceiling* are best understood with the following drawing:

2+-"Uns
floor

•: -+--*ceiling*
-1.6
-2 -+--*floor*

Relationship
of *ce;fing* and *floor*

Just think of the *ceiling* as it is in a house.. on top. Likewise, think of the *floor* as being on the bottom.

Therefore, *Math.ceil(-1.6)* being *-J* makes perfect sense since *-J* is above. Similarly, *-2* is below *-1.6* so it makes sense to say that -2 is *Mathjloor(-1.6).*

8. double d = 7.89;
System.out.println(Math.log(d)); //2.065596135 ...log is base e.

9. double x = 2038.5;
double y = -8999.0;
System.outprintln(Math.min(x,y)); //-8999.0

IO. double x = 2038.5;
double y = -8999.0;
System.outprintln(Math.max(x,y)); //2038.5

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

double x = 148.7;
System.outprintln(Math.round(x)); //149

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

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

12. Systemout.println(Math.PI); //3.14159265...

Advanced *Mat/1* methods:
Below are .some additional *Math* methods that advanced math students will find useful:

Method	Signature	Description
log	double log(double x)	Returns log base e of x
sin	double sin(double a)	Returns the sine of angle a... a is in rad
oo,	double cos(double a)	Returns the cosine of angle a... a is in rad
Ian	double tan(double a)	Returns the tangent of angle a... a is in rad
a<in	double asin(double x)	Returns arcsine of x...in range-PI/2 to PV2
aoo'	double acos(double x)	Returns arccosine of x...in range 0 to Pl
atan	double atan(double x)	Returns arctan of x. in range -PI/2 to PI/2
toDegrees	double toDegrees(double angRad)	Converts radians into degrees
toRadians	double toRadians(double angDeg)	Converts degrees into radians

Exercise on Lesson 6

L 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?
int k = Math.abs(-127.5);

4. Write a statement that will print the result of 2^{15}.

5. System.outprintln(Math.ceil(-157.2));

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

7. System.out.println(Math.ceil(l57.2));

8. System.outprintln(Math.floor(157.2));

9_ System.outprintln(Math.round(-157. 2));

10. System.outprintln(Math.ceil(-157.7));

1L System.outprintln(Math.ceil(l57));

12. System.outprintln(Math.ceil(157.7));

13. Write a statement that will print the natural log of 18.. . same *as* ln(18) on a calculator.

14. Write a line of code that multiplies *double p* times 1!and stores the result in *h.*

Project ..• Compute This

Create a new project called *ComputeThis* having a class called *Tester*. The *main* method of *Tester* should calculate the value of the following formulas and present the answers as shown.

dl =3nsin(187°)+lcos(122°)1 ...Remember that the argmnents of sin and cos must be in radians.

d2 =(14.72)$3^{"801}$ + In 72 ..In means log base e

The output of your code should appear as follows:

```
dl     -0.618672237585067

d2     27496.988867001543
```

Verify these answers with a calculator.

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 *nextlnt* method to input an integer from the keyboard:

```
import java.io.*;  //see "Imports necessary" on next page
import javautiL*;
public class Tester
{
        public static void main( String argsO )
        {
                Scanner kbReader = new Scanner(System.in); //see "Mysterious
                                                        //objects" on next page
                System.out.print("Enter your integer here. "); //enter 3001
                int i = kbReader.nextlnt( );
                System.outprintln(3 * i);  //prints 9003
```

Inputting a *double:*

Use the *nextDouble* method to input a *double* from the keyboard:

```
import java.io.*;
import java.util.*;
publie class Tester
{
        public static void main( String args[] )
        {
                Scanner kbReader = new Scanner(System.in);
                System.out.print("Enter your decimal nwnber here. '"); //10005
                double d = kbReader.nextDouble( );
                System.outprintln( 3 * d );  //prints 3001.5
```

Inputting a *String:*

Use the *next* method to input a *String* from the keyboard:

```
import java.io.*;
import java.util.*;
public class Tester{
        public static void main( String args[] )
        {
                Scanner kbReader = new Scanner(System.in);
                System.outprint("'Enter your String here. ");//Enter One Two
                String s = kbReader.next( ); //inputs up to first white space
                System.outprintln( "This is the first part of the String,... "+ s);
                s = kbReader.next( );
                System.outprintln{ "This is the next part of the 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
```

Multiple inputs:

In a similar way *nexJlnt()* and *nextDouhle()* can be used multiple times to parse data input from the keyboard. For example, if 34 88 192 18 is input from the keyboard, then *ne.xt/nt()* 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( ):*

```
importjava.io.*;
import java.util.*;
public class Tester
{
    public static void main( String argsQ )
    {
        Scanner kbReader = new Scanner(System.in);
        System.outprint('"Enteryour 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 my string, ... One Two
```

Imports necessary:

We must import two classes,...*Java.io. *andjava.util. ** that provide methods for inputting integers, *douhles,* and *Strings.* See Auuendix I for more on the meaning of "importing".

Mysterious objects:

In the above three examples we used the following code:

Scanner kbReader = new Scanner(System.in);

It simply creates the keyboard reader object (we aihitrarily named it *khReader)* that provides access to the *nexllnl(), nextDouble( ), next( ),* and *nextLine( )* methods. For nowjust 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 tol.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 *nextlnt(), nextDouhle( ), next(),* and *nextLine( )* may be used in any sequence with the following exception:

It is not permissible to follow *nexllnt( )* or *nextDouble( )* with *nextLine( ).* lf it is necessary to do this, then a new *Scanner* object must be ronstructed for use with *nextLine( )* and any subsequent inputs.

Project... Going in Circles

The area of a circle is given by:

area = it (i'-)

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

Write a program (project and class both named *RadiusOjCircle)* 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 th.em 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 Operatcirs

Back in Lesson 2 we looked at three fundamental variable types... int, *douhle,* and *String.* Here, we look at another very important *type.....hoolean.* 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?

((x <10) AND (y = 97)) Both parts are *true* so the whole thing is *true.*

((x <10) AND (y =-3)) First part is *true,* second part is*false,* whole *thingfalse*

((x <10) OR (y = 97)) If either part is *trne* (both are) the whole thing is *true.* (

(x <IO) OR (y =-3)) If either part is *troe* (first part is) the whole thillg *troe.*

Correct syntax:

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

1. To compare two quantities...such as (y = 97) above we must instead do it this way:
   (y == 97). ..recall that a single"'=" is the assignment operator.

   Similarly, read y != 97 as "y is not equal to 97".

2. In Java we don't use the word "and" to indicate an AND operation as above. We use "&&" instead........((x <10) && (y = = 97))

3. InJava we don't use the word "or" to indicate an OR operation as above. We use "II" instead........((x <IO) 11 (y = = 97))

Truth tables:

Here are truth tables that show how && and II work for various combinations of *a* and *b:*

		.,;:q:
false	ful"	false
false	true	fiili;e
true	fulse	false
true	true	true

Table 8-1 AND-ing

		'
false	fulre	fulse
false	true	true
true	ful"	true
true	true	true

Table 8-2 OR-ing

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 ifwe say not true (!true)? ... false, of comse.

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

Creation of *booleans:*

Create *boolean* variables as shown in the following two examples:

boolean b = true;
boolean z = ((p <j) && (x != c));

Use the following code for example 1-10 below:

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(c 1 1 (d > 0)); //true

4. System.out.println(!b II c); //false

5. System.out.println((x >102) && true); //false

6. System.out.println((jj = = 1) 11 false); //false

7. System.out.println((jj==40) && !false); //true

8. System.out.println(x != 3); //true

9. System.out.println(!(x!=3)); //false

10. System.outprintln(!!true); //true

Operator precedence:

Consider a problem like:

System.outprintln((true && false) 1 1 ((true && true) 11 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 II 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 Hfor a complete listing of operator precedence). The order is:

!= &&

Example 1

System.outprintln(true II false && false); //true

Do the false && false part first to get a result of false.
Now do true II false to get a final result of true.

Example 2

```
System.outprintln(   true  && false || false);  //false
```

Do the true && false part frrst to get a result of false.
Now do false II 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 insomething like,

"Java scripf' and "Bill Gates"

and you will find only references that contain both these items.

On the other hand, enter something like,

"Javascripf'or"BillGates"

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.

Now that we have learned to write a little code, it's time to tum 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 Aonendix S.

Exercise for Lesson 8

In problemsl -5 assume the following:

```
int z =23, x =-109;
double c = 2345.19, v = 157.03;
boolean a = false, s = true;
```

I. boolean gus = (x > O) && (c = = v);
System.out.println(!gus);

2. System.out.println(a 1 1 s);

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

4. boolean r = z = =x;
System.out.println(r 1 1 false);

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

6. FiU in the following charts.

a	b	(!a && b)
fuho	false	
fuho	truo	
truo	IIIII	
truo	truo	

a	b	(a \ I !b)
false	false	
false	truo	
truo	false	
truo	truo	

7. Assume *b*,*p*, and *q* are *booleans.* Write code that will assign to *b* the result of AND-ingp 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 h are integers. Store the result in *boolean k.Dog.*

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) 11 ((true && true) 1 1 false));

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

16. System.outprintln(true || true && false);

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

18. SysteITLout.println(false && true || false);

Lesson 9...•The *if* Statement

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

Example 1:
```
//Get a grade from the keyboard
Scanner kbReader = new Scanner(System.in);
System.out.print{"What is your grade? ");
int myGrade = kbReader.nextlnt( );

//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.outprintln("Congratulations,   you  passed.");

else
{
        //Execute  code here ifthe  test above is false
        System.outprintln("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:
```
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 = O;
if(  (state = = "Texas") || (state = = "Tx")  )
{
        //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 + ".");
```

Itwon't work!

There is just one dlfficulty with the above code in Example 2. Itwon't work! The problem is with how we are trying to compare two *Strings.* It cannot be as follows:
```
state = = "Texas"
```

Rather, we must do it this way:
```
state.equals("Texas")
```

A good way to cover all the bases in the event someone mixes upper and lower case on the input is as follows:
```
( state.equalslgnoreCase("Texas")  J | state.equalslgnoreCase("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:
```
int groovyDude = 37;
if (groovyDude = =37)
        groovyDude++;  //this line.!! executed iftest is true
System.out.println(groovyDude );  //38
```

Example 4:
```
int groovyDude = 105;
if (groovyDude = =37)
        groovyDude++;  //this line is not executed iftest is false
System.out.println(groovyDude); //105
```

The *else* if:

Multiple ifs can be used in the same structure using *else if.*

Example 5:
```
//Get a grade from the keyboard
Scanner kbReader = new Scanner(System.in);
System.out.println("What is your grade? ");
int theGrade =kbReader.nextlnt(  );

if (theGrade>=90)
(
        System.outprintln("You made an A.");
}
else if (theGrade>=80)
{
        System.outprintln("You made a B.");
}
else if (theGrade>=70)
{
        System.out.println("You made a C.");
}
else if (theGrade>=60)
{
        System.outprintln("You made a D.");

else

        System.outprintln("Sorry, you failed.");
```

Exercise on Lesson 9

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

```
int i = 10,j =3;
boolean true_false;
```

I. true_false =G>i);

2. true_false =(i >j);

3. true_false =(i= =j);

4. true_false = ((j<= i) 1 1 G >= i));

5. true_false = ((i >j) && G = = O));

6. true_false = (G < 50) 1 1 G !=33));

7. true_false = (!(j >= 0) 1 1 (i <= 50));

8. true_false = (!(!(!true)));

9. true_fu.lse = (5 <= 5);

10. true_false = (j!= i);

1L Write a statement that will store a true in *boolean h* if the value in the variable m is 44 or less.

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

13. What is returned by the following expression? (Recall thatthe precedence order of logical operators is !, &&, and finally 1 1-)
!{ {2>3) 1 1 {5==5) && (7>t) && {4<15) 1 1 (35<=36) && (89!=34))

Inproblem 14–16 what is the output?

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

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

16_ intj = 19, m = 200;
```
if G= =I8)
     m++;
     j++;
System.out.println(m);
SysteIILout.printlnG);
```

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

18. Write a statement that will store a *true* in *boolea11 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". Ifit's entered exactly as "XR.ay", printout "Passwoi'd entered successfully.";- otherwise,, have it printout "Incorrect password."

20. What is output by the following "nested ijS" code?
```
int k = 79;
if (k>50)
{
        if (k<60)
        {System.out.println("One");}
        else
        { System.out.println("Two");}

else
{
        if (k>30)
                System.out.println("Three");
        01'0
                System.out.println("Four");
```

Project...Even or Odd?

Create a new project called *EvenOrOdd* containing a class called *Tester*. In the *main* method of *Tester* print a prompt that says, '"Enter an integer:" Input the user's response from the keyboard, test the integer to see if it is even or odd {use the modulus operator ob to do this), and then print the result as shown below {several runs are shown).

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

Enter an integer; 2049
The integer 2049 is odd.

Enter an integer: -236
The integer -236 is even.
```

Keys **for** Quizzes/Exercises/Projects

The short quizzes for each lesson inthis section are not comprehensive and not very difficult Normally, only basic, superficial questions are asked. The general philosophy here is for the specter of a quiz to always be hanging over the student where he knows he must quickly acquire a general working knowledge of the subject but at the same time knows he will not be asked in-depth or tricky questions. It is hoped that this gentle, but persistent pressure, will encourage the student to keep current with his studies and be rewarded with a frequent "100" on these little quizzes. It is suggested that a quiz be given **the day after** a new lesson is introduced.

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:

```
System.oulprint("Fire");
System.out.println(" Ants");
```

Which of the following is actually printed?

a. Fire Ants

b. Fire
An1'

4. What is the syntax for indicating that a line of text is not Java code; rather, it is arem?

Quiz on Lesson 2

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

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

4. Which of the following are illegal names for a variable?

3. Write a line of code that declares *k* to be fill integer.
   num ,Nlllll, count, 12flag. flag-stuff, flagStuff; flag_stuff; flag stuff

5. Which of the follo\Ving 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 tQis 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 , NUIIL. count, 12fl,, flag-stuff; flagStuff; flag_stuff, flag stuff

5. Which of the follo\Ving is the least desirable way to name a variable?
   redcolor, red -color, redColor

Key to Exercise **on** Lesson 2

L What are the three main types of variables used in JAVA: and.-:;vhat-are tl\ey used to store?
a. String•..stores names, letters, sentences, o'r 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?
donble•••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 1.921×10^{-16} in it.
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
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.
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.
int count;

9. Write a line of code that initializes the double precision variable *bankBalance* to 136.05. Assume this variable has already been declared.
bankBalance = 136.05;

IO. 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; //legal
int alt = 1493.86; //illegal
 int num = 1_2_3; //legal
```

Quiz on Lesson 3

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

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

3. What is output by the following code?
```
Sning girl =""Heather Jones";
System.out println(girl.substring(8, 11});
```

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

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

9. Write code that will-produce the following printout using only, a.single *println(/*,

```
Hello
Hello  again
        System.out.println("Hello\nHello again");
```

10. Write code that will produce the followlng printout.

```
A backslash looks like this \, ...right?
        System.out.println("A backslash looks like this \\, .•.right?");
```

11. What is output by the followlng?

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

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

●

~~**Project... Name That Celebrity, Key**~~

```
public class Tester
{
        public static void main(String args[])
        {
                String s1 = "Allan Alda";
                String s2 = "John Wayne";
                String s3 = "Gregory Peck";

                int len = s1.length( );
                String s = s1.substring(2, len - 3);
                System.out.println(s1 + ">>>" + s);

                len = s2.length( );
                s = s2.substring(2, len - 3);
                System.out.println(s2 + ">>>" + s);

                Jen = s3.length( );
                s=s3.subs1ring(2,Jen-3);
                System.out.println(s3 + ">:>>" + s);
        }
}
```

Quiz' on Lesson 4

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

2. What is another way to write *p =p-1;* ■

3. Write.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 intj* = J;what will be the value *ofj+* +?

2. What is another way to write *p =p-1; ?*

3. Write *.x* ±=*j;* another way.

4. Write codilthat-ilil'lvru e.ying the new value of w is the old value of w plus 6.

5. Which of the fi llo · g 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 de would not compile:

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

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

3. double def,
double f = 199237;
def = f,
System.outprintln(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 + l); //3

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.
v-=p-30;

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

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

9. int sd = l2;
int x = 4;
SysteIILout.println(sdo/o(++x)); //2
System.out.println(x); //5

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

11. Ona 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;
intj = 5;
m = m / j; // new m is old m divided by j
System.out.println(m);
What's printed?
7

14. Systemout.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 *i* by*j* in a variable named *k*
k= i%j;

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

18. Show three different ways to decrement the variable}.
j-; -j; j=j-1;

Project... Cheating on Your Arithm t" Assignment

```
r1bhc class Tester

        public static void main (Stnng '' [])
        {
                int answ =7 3 * (182 - 68) -7 + 19;
                System.out.;ri(".+ 3 * (4 + 82 - 68) -7 + 19 = "+ answ + "\n"),

                answ = (179 + 21 + 10) / 7 + 181;
                System.out.println("(179 + 21 + 10) / 7 + 181 =  " + answ + "\n");

                answ = 10389 * 56 * 11 + 2246;
                System.out.println("10389 * 56 * 11 + 2246 = " + answ);
```

Quiz on Lesson 5

1. Which of the followingis 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 itlegal
   ```
   double d = 187.2;
   int j = d;
   ```

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

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

Key to Exercise on Lesson 5

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

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 numbe"r of students, *NUM STUDENTS*, to 236.
   final int NUM_STUDENTS = 236;

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

4. int cnt = 27.2; //Won't even compile. This line is illegal
   System.out.println(cnt);
   What's printed?

5. double d = 78.I;
   int fg = (int)d;
   System.outprintln(fg); //78
   What's printed?

6. Is double f4 = 22; legal? Yes

7. The following code stores a 20 in the variablej:
   doublej = 61/3;
   What small change can you make to this single line of code to make it produce the "real" answer to the division?
   doublej =(double)61/3; or doublej =61.013;

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

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

10. int p = 3;
    double d = 10.3;
    intj = (int)5.9;
    System.out.println(p + p * d -3 *j); *1118.9*

11. int p= 3;
    double d = 10.3;
    intj =(int)5.9;
    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 = O;
int dividend2 = 13, divisor2 = 3, quotient2 = O. remainder2 = O;
quotient = dividend/divisor,
remainder = dividend % divisor,
quotient2 = dividend2 l divisor2;
remainder2 = dividend2 % divisor2;
```

12. System.outprintln(quotient); //3

13. System.outprintln(remainder); //0

14. System.outprintln(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 *douhle* so that strictly integer division is done. Store the result inj, an integer.
    intj = (int)d *l 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. ls the following code legal? If so, what is printed? If not, why?

    int k = 7;
    k*=.5;
    System.outprintln(k:);
    yes, legal
    3 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?*
   System.out.println(Math.ceil(d));

4. Write a line of code that will calculate $(3.1)4^{\cdot 052}$ and store the result in *double d.*

Key to Quiz on Lesson 6

1. Show how we wll1 raJculate and print the square root of 139.46.

2. Show how we would print the value of 1t.
   a a

3. Whai would the following print if *d =208.4'!*
   System.ou rintln(Math.ceil(d));

4. Write a line of code B will calculate f3.1)$4^{\cdot 052}$ 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 iny.
```
double y=Math.sqrt(x);
```

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

3. Is the following legal'? If not, what would you do to make it legal'?
```
int k = Math.abs( 127.5);
```
No, it tries to store a *double* in an *int.* There are two things you could do:
```
double k=Math.abs(-127.5);
    O .
int k=(int)Matb.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^{15}.
```
System.out.println( Math.pow(2, 1.5));
```

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

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

7. System.oulprintln(Math.ceil(157.2)); //158.0

8. System.outprintln(Math.floor(157.2)); //157.0

9. System.outprintln(Math.rotmd(-157.2)); //-157

10. System.outprintln(Math.ceil(-157.7)); //-157.0

11. System.outprintln(Math.ceil(157)); //157.0

12. System.outprirrtln(Math.ceil(157.7)); //158.0

13. Write a statement that will print the natural log of 18 ... same as ln(l8) on a calculator.
```
System.out.println( Math.log(18) );
```

14. Write a line of code that multiplies *double p* times 1t and stores the result in *b.*
```
double b =p * Math.PI;
```

, ,?

Project•.. Compute This

```
public class Tester
{
   public static void main(String args[])
   {
            double dl =3*MaifP1 .  atb.sin(Math.toRadians(187)) +
                          >f            Math.abs( Math.cos(Math.toRadians(l22)) );
            System.out.printf .("dl  = "+    + "\n");
            double d2 = Ma th.pow{14.72, 3.   1) + Math.log(72);
            System.out pr ntln("d2  = " + d2);,
```

Quiz on Lesson 7

l. 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 kb*Reader* 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, *khReader*, 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

l. What are the tluee 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 *khReader* to input a quantity like "Jo mamma" from the keyboard and store the result in the variable, *name.*

4. Suppose a *Scanner* object, *khReader*, 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 II *'!*

```
System.outprintln(bb);
```

6. What is the output of the following code assuming that p is *true* and *q* is *false?*

```
boolean bb = !p II 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?

[redacted]

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

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

5. Which hDher precedence && or II ?

```
System.outprintln(bb );
```

6. What is the output of the following code assuming that p is *true* and *q* is *false?*

```
boolean bb = !p II q;
```

Key to Exercise for Lesson 8

In problems I-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 > O) && (c = = v);
   System.out.println(!gus);   I/true

2. System.outprintln(a 1 1 s); //true

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

4. boolean r = z = =;
   System.outprintln(r 1 1 false); //false

5. System.outprintln(z!=x); //true

6. Fill in the following charts

a	b	(!a && b)
ful<O	ful<O	false
false	tru'	trno
tru'	ful"	false
tru'	tru'	false

a	b	(a 1 1 !b)
false	false	truo
ful<0	[illegible]	filio
tru'	false	[illegible]
truo	truo	trno

7. Assume *b*,*p*, and *q* are booleans. Write code that will assign to *b* the result of.AND-ingp and *q*.
   b = p && 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 ify equals z:
   w = (x > O) !l (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*a* is not equal to *b*. Assume *a* and b are integers. Store the result in *boolean kDog*.
    boolean kDog = a l= 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? 1 1

·14. System.outprintln((true && false)] I ((true && true) 1 1 false)); //true

15. System.outprintln(true && true II false); //true

16. System.outprintln(true II true && false); //true

17. System.outprintln(false II true && false);//false

18. System.outprintln(false && true II false); //false

Quiz on Lesson 9

1. Show the basic skeleton of an *if· else* structure.

2. Inthe following code, assume that the portion designated with **<#1>** is a *true* st.atement What will be the output?

```
if( <#1> )
{
        SysteIILoutprint("Elvis");
}
System.out println("' Presley");
```

3. Inproblem 2, what would be the output if <#l> was a*false* statement?

4. Ifyou wanted to compare the contents of two *Strings, sl* and *s2,* which of the following would be appropriate to use?
   a s1.equals(s2)
   b. s2.equals(sl)
   c. sl.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.

[redacted]

2. In the following code, assume that the portion designated with **<#1>** is a *true* statement. What will be the output?

```
if( <#1>)
{
        System.out.print(""Elvis");
}
System.outprintln("' Presley");
```

[redacted]

3. Inproblem 2, what would be the output if <#l> was a*false* statement?

[redacted]

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

[redacted]

Key to Exercise on Lesson 9

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

```
int i "' 10,j =3;
boolean true_false;
```

L true_false = G > i); J/false

2. true_false = (i > j); //true

3. true_false = (i= =j); //false

4. true_false = (G <= i) l l G >"' i)); //true

5. true_false "' ((i > j) && G = = O)); //false

6. true_false = (G < 50) 1 1 G != 33)); //true

7. true_false = (!G >= 0) 1 1 (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 troe* in *boolean b* if the value in the variable *m is* 44 or less.

    boolean b = m <F- 44;

12. Write a statement that will store a *false* in *boolean b* 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 11.)

    !((2>3) 1 1 CS 5) && (7>1) && (4<15) 1 1 (35<36) && (89!34))

    false

In problems 14 16 state what's output.

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

15.
```
String sl ="school BUS";
if( sl.equalslgnoreCase("school bus"})
System.outprintln("Equal");
else
System.outprintln("Not equal");
```

16_
```
intj = 19, m = 200;
if G= =l&)
        m++;
        j++;
System.out.println(m); //200
System.outprintln(j}; //20
```

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

    boolean b = g = =34;

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

    boolean b = (k%2) = = O;

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.print("Enter your password. ");
String passWord = kbReader.nextLine( );

if( passWord.equals("XRay") )
{
    System.out.println("Password entered successfully.");
' '

    System.out.println("lncorrect password.");
```

20. What is output by the following "nested *if'* code?

```
int k = 79;
if (k>50)
{
    if (k<60)
     {System.out.println("One");}
    else
    {System.outprintln("Two'");}
)
else

    if (k>30}
        System.out.println("'Three)";
    else
        System.out.println("Four");
```

Test Through LCssO:ri-3

1. -Write the single line o_f code that will cause Computer cience to be printed on the screen.

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

   Helloworld

3. Suppose you want to document yom program with the following text

   Program author: Miss Fefe LaFu

   ·Modify this line of text so that it can legally reside in yom program as a·comnient

4. What does *rem* stand for?

5. The following bl9ck of text is to be considered a remark Rather than putting a double slash in front of each Ille of text, mcidify the text using the "block" technique of "commenting-out" the text.

   Name: Elvis Presley
   Date:Jan 34, 1776
   School:.Nashville High School

6. Write the following nuinber in a more conventional way as Otild be done in a math class.

   7_80993£-12

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

   int siv;
   siv = 14;

8. Suppose we wish to cfeate avariable name using the words diStanCe to lifie". Which of the following is the conventionally acceptable way to name this variable? (There may be more than one answer.)

DistanceToLine	distanceToLine	disc·to·'Hne
Distance_To_Line	distancetoline	distanceTo-Li'e
DISTANCETOLINE	distanceTOline	distance-to-lini:

9. Which of the following are legal variable names regardless of whether they are conventionally acceptable?

Arrow	public	integer	double	came·ra
bOokX	x	xx	dinner bell	dmD.e'rben
dinnibell	XCORD	myName	23Yg	dJ9"

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

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

12. What's wrong (if anything) with this line of code?

    Int ii = 59;

13. What's wrong (ifanything) with this line of code?

    double dur = 102;

14. What's wrong (ifanything) with this line of code?

    int pugh = 1003.84;

15. What's wrong (if anything) with this line of code?

    String myNumber = "122.809";

16. Rewrite the following line of code as two separate Jines of code.

    double dd = 1000.56;

17. Suppose you have two Strings, *strl* and *str2.* Write a line of code that concatenates these two strings (with *str2* comlllg first) and assign the result to the *String p.*

18. Suppose that 17 above *was* done correctly. What would the following print assuming that strl = "Hello" and str2 = "world"?

    System.out println(p);

19. Usingtrl and *str2* as defined in 17 and 18 above, concatenate them and assign to *String h* so that the result will be "Hello world".

20. What is output by the following code?

    String s = "My dog";
    int theLen = s.length( );
    System.out.println(theLen);

21. What is output by the following code?

    String gg = "Wiggle worm";
    String s = gg.subsning(3);
    System.outprintln(s);

22: What is output by the followmg code?"

```
String gg = "Big bad }VOlf';
String s = gg.substring(2,6);
Systemout.P;rintln{s);
```

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

24. What is output by the following code?

```
String xc = "Big Wally";
System.outprintln( 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 We **have "bad" enough.**

27. What is output by the following code?

```
String pgr = "ijklm\nopqrs";
System.outprintln(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, **The String bas 22 characters.** The 22 part is just an example. Use the *length()* method to print the 22 (or whatever it might be actually be for v).

30. What will be the value *ofjy?*

```
String bj = "Oh my goodness!"
int pl = 4;
Stringjy = bj.substring(O, pl };
```

31. What will be the output of the. following code?

```
String s = "Kitty cat";
int kc_Jen = s.length( );
String ms = s.substring(kc_len-2);
Systemoutprintln(ms);
```

32. What is the meaning of "concatenatin,.., furo-'sfrib:gi tb· ilie? -·

33. Give the names of two methods of the *String* class.

Test Through Lesson 7 (Emphasizing 4-7)

I 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:
```
Scanner kbReader = new Scanner(System.in);
System.outprint("What is your code? ");
<#1>   //this line of code stores in m whatever is typed in 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? (Assmne that the data to be input is always just digits as in the example.)
   a String m = kbReader.nextLine( );
   b. int m = kbReader.nextlnt( );
   c. double m = kbReader.nextDouble( );
   d. All of the above
   e. None ofthese