Beginning Programs

All Java programs have similar structures, much like a house. Your house will have roof, a foundation, bedrooms, bathrooms, a kitchen and so on. Your house may be small and your neighbor’s house large, but they still have the same structures. The following is an example of a simple Java program:

```java
public class sample {
    public static void main(String[] arg) {
    }
}
```

This is a valid program that does absolutely nothing. For now, accept that all programs begin with “public class .”. The name of the program, in this case, `sample`, should reflect the purpose of the program. The curly brackets ({  }) mark the beginning and end of the entire program (sample) and the main function. Also for now, accept that all of our program code will go between the curly brackets of the main function.

**Comment Statements:**
Adding remarks, or comments, to your programs is good programming practice. Comments in small simple programs may appear to be unnecessary, but as your programs grow, commenting will make them easier to follow and debug. In industry, commenting is a must. There are two kinds of comment statements: single-line comment and a block comment.

```java
//This is a one-line comment statement.
//Author: Debbie Gamel
//Everything after the // ON THIS LINE is not executed.

/* This is a block comment. Everything after the /* is not executed. The compiler will not check for errors or compile again until it reaches the end of the block. The end of the block is */
```

**Output: System.out**

```java
System.out.println("Gamel");       //prints the Gamel to the screen with cursor on same line after the ‘l’
System.out.println("Gamel");       //prints Gamel with cursor on next line
```

You can advance a line by doing `System.out.println();`;

Notice that the statements end with a semicolon. **Most** Java statements end with a semicolon, much like a period terminates a sentence.

**Escape Sequences:**
An escape sequence is used to print a special character. For example, if you need to print a quote, the compiler gets confused: `System.out.println("\"\")`; What does it do with the third quote? The first quote marks the beginning of what is to be printed and the second marks the end. An escape sequence is the solution. When the compiler encounters a backslash, it knows to “look up” what is needed, like a mini-command.

```java
\t:  //advances to the next tab stop
    System.out.println("\tG\ta\tm\te\tl");

\n:  //advances a new line
    System.out.println("\nGamel\n");

\\:  //prints a back slash

\":  //prints a double quote

\':  //prints a single quote
```

Gamel: CS1A
Shortcut

The output statements get long and repetitive, so if you would like a shortcut, we can include a library that allows this:

```java
import static java.lang.System.*;
```

Now the statements above can be: `out.print("Gamel");` or `out.println("Gamel");`

Since output is used throughout a program, including the library once to use the shortcut many times over saves us keystrokes.

```java
import static java.lang.System.*;
public class outputGamel {
    public static void main(String[] arg)
    {
        out.println("\nMy name is Debbie Gamel");
    }
}
```

Imports

Java is arranged into libraries, which contain packages and classes. See the diagram below:

In the diagram to the left, the outer box represent a java library. The small rectangles represent java packages and the inner circles represent java classes. Code in java.lang (java library and lang package) are available at all times. To use classes from any other package, we must fully qualify its location using the import statement.

Thus the shortcut above,

```java
import static java.lang.System.*;
```

allows us to use the commands in the java library, lang package and System class.
Java Output Programs:

TO BEGIN:
1. Open J Creator.
2. File / New / File
3. Click Java Classes (left column) / empty java file (right column) then next.
4. Type in the program name (for example: NAX2). Do not type .java.
5. Click the three dots button beside C: Go to your H drive, Unit_2_BeginningJava\SimpleOutput.
6. Click Finish.

Output Program assignments

Name and address times 2:
Save as: NAX2.java:

Write a program to output your name, address, and city / state with each on a different line using a separate command for each line of output. Print two blank lines. Now do the exact same output using only one command. Be sure to have a println() command at the end of the program so “Press any key to continue” does not smoosh next to your output. Sample run on next page.

Example run for NAX2:

Joe Smith
123 Avenue A
Katy, TX 77450

Joe Smith
123 Avenue A
Katy, TX 77450

Star Wars Name:
Save as: starWarsName.java

Write a program that will print your name, a sentence stating the city of your birth, a sentence stating your mother’s maiden name and your Star Wars name. To find your Star Wars name do the following:

First name: The first 3 letters of your last name + The first 2 letters of your first name.
Last name: The first 2 letters of your mother’s maiden name + The first 3 letters of your birthplace.

Lastly, separate with a blank line. Next print your Star Wars name again, with each character separated by a tab ("\t").

Example Run:
My name is Debbie Gamel.
I was born in Omaha, NE.
My mother’s maiden name is Willett.
My star wars name is Gamde Wioma.

G a m d e W i o m a
FORMATTING in COLUMNS

Although not on the AP exam, another output command is used for formatting. We will learn additional options as we proceed. The command is:

```java
out.printf("%-10s %-15s %-5s %-10s", "xxxxx", "yyyyyyyyyy", "z", "xyz");  // The f stands for format.
```

This could also be written as:

```java
out.printf("%-10s", "xxxxx");
out.printf("%-15s", "yyyyyyyyyy");
out.printf("%-5s", "z");
out.printf("%-10s", "xyz");
```

The formatting commands are always inside quotes. The % means special commands follow. The minus sign means the string is left justified; if omitted, the string will be right justified. After the %, designate the width of the column or output field. Last, the s specifies that string data will go in the column. (Other data types introduced soon.) Notice that there are no commas separating each of the fields of data. The spaces above are not required and actually add a space in the output field. You will have %-XXs for every data item being output on one line. Last, "xxxxx", "yyyyyyyyyy", "z", and "xyz" are the actual texts that will be outputted on your monitor. If you have several lines in column format, the formatting commands will be identical, but the data will change.

The printf command will not be on your tests or quizzes but you will be required to use it on some programming assignments.

**Schedule Format Program:**

**Save as Schedule.java**

Write a program that prints your schedule on the screen. You should line up your class, period and teacher.

**Example run of Schedule:**

<table>
<thead>
<tr>
<th>Period</th>
<th>Class</th>
<th>Teacher</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Algebra</td>
<td>Sammons</td>
</tr>
<tr>
<td>2</td>
<td>Computer Science</td>
<td>Gamel</td>
</tr>
<tr>
<td>3</td>
<td>Chemistry</td>
<td>Modic</td>
</tr>
<tr>
<td>4</td>
<td>English</td>
<td>Srack</td>
</tr>
<tr>
<td>5</td>
<td>History</td>
<td>Hunt</td>
</tr>
<tr>
<td>6</td>
<td>Art History</td>
<td>Milton</td>
</tr>
<tr>
<td>7</td>
<td>Football</td>
<td>Joseph</td>
</tr>
</tbody>
</table>

Gamel: CS1A
Declarations and Data Types

Java has two different categories of data types: primitive and classes

The following are the **primitive** data types:

- **int**: Contains integers such as 7, 12, -5; int is used whenever you will not need decimal values. When referring to the contents of a variable that contains an int, there will never be quotes – just the value.

- **double**: Contains numeric values such as 4.2, 12.9999955, 1.0, -5.4; double is used whenever you will need a decimal point. Again, when referring to the contents of a variable that contains a double, there will never be quotes – just the value.

- **char**: Contains characters such as ‘a’, ‘T’, ‘*’, ‘?’, ‘5’; char is used to hold any single character. Note that ‘5’ (a character 5) is different from 5 (an integer 5) which is also different from 5.0 (a double 5). When referring to the contents of a variable that contains a char, there will be single quotes.

- **boolean**: Contains true or false

Classes are different from primitive data types. A class is a data type that holds a reference to (the address of) an object. For example:

```java
bug bob = new bug();  // bob contains the address of a bug object
card c1 = new card();  // c1 contains the address of a card object
```

A variable created using primitive data types holds the actual data. An object created using any class contains the address of the data. The class you will use most often at the beginning of the course is the String class. For now, you need to know the difference between a class and primitive data type, but we will study classes more in depth later.

- **String**: Contains strings such as “Mrs. Gamel”, “Computer Science”, “a”.
  Note that ‘a’ is a character `a` and “a” is a string `a`. When referring to the contents of a variable that contains a string, there will be double quotes.

- **Scanner sc = new Scanner (System.in);**

Given the following values as contents of a variable, label them as int, char, boolean, double or String.

- 9
- Wilson
- +
- 27.22
- 3.14159
- Java Solutions

Gamel: CS1A
Declaring Variables:

When declaring variables, you must assign a name to the variable, following the rules below:

- Names must begin with an alphabetic character or underscore (_).
- You may use alphabetic or numeric characters in the variable name.
- The **only** special character allowed is the underscore. **(NO SPACES, COMMAS, PERIODS, SEMICOLONS, COLONS, QUESTION MARKS, FORWARD OR BACK SLASH, ETC.)**
- No reserved words are allowed. *(For example, you cannot name a variable “public”.)*
- No two variables in the same method may share a name.

Declaration Examples:

```java
int x;  // Declare an integer variable named x.
double hourlyRate;  // Declare a double variable named hourlyRate.
String firstname;  // Declare a string variable named firstname.
```

You can also declare and initialize in one statement.

```java
int age = 16;  // The same as:
age = 16;  // Declare and initialize age to 16.
```

The first one is declaring age to be of type `int` **and** initializing age to 16.

**Check your understanding of variable names:**

State whether the variable names are valid (V) or invalid (I).

1. V City
2. V student1
3. V Parent’sName
4. V 1student
5. V Number of Miles
6. V student_1
7. V NumberOfMiles
8. V Sales%rate
9. V NaMe
10. V USERNAME
11. V Gamel
12. V highest sales
13. V last name
14. V FirstName
15. V Last_name

Gamel: CS1A
Data Types and Variable Names Homework

State whether the variable names are valid (V) or invalid (I).

1. __________ age
2. __________ StReEt
3. __________ tax rate
4. __________ Last Name
5. __________ S_S_#
6. __________ GPA
7. __________ MY_address
8. __________ A
9. __________ total/count
10. __________ String
11. __________ 999counter
12. __________ counter999
13. __________ SS_Number
14. __________ WhoAreYou?
15. __________ COUNT
16. __________ Stud_Average
17. __________ weight
18. __________ U.S.A
19. __________ x-y
20. __________ void

Given the following variable names, specify the correct data type.
Examples: int age and char letterGrade

21. __________LastName
22. __________ age
23. __________TaxRate
24. __________middle_initial
25. __________GPA
26. __________SS_Number
27. __________address
28. __________gender //(M or F)
29. __________Course
30. __________employed
   //(true or false)
31. __________school_name
32. __________NumberOfStudents
33. __________Hourly_Rate
34. __________SrCitizen
   // (true or false)
35. __________numMiles
36. __________PhoneNumber
37. __________alphaLetter
38. __________author
Answer Key: Data Types and Variable Names HW

1. V
2. V
3. I
4. I
5. I
6. V
7. V
8. V
9. I
10. I
11. I
12. V
13. V
14. I
15. V
16. V
17. V
18. I
19. I
20. I
21. String
22. int (or double if partial years are important, such as for infants or young children)
23. double
24. char
25. double
26. String
27. String
28. char
29. String
30. boolean
31. String
32. int
33. double
34. boolean
35. int (or double if partial miles is significant)
36. String
37. char
38. String
## ALL DATA TYPES

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Memory Usage</th>
<th>Min .. max</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Integers:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>byte</td>
<td>8 bits</td>
<td>-128 to 127</td>
</tr>
<tr>
<td>short</td>
<td>16 bits</td>
<td>-32768 to 32767</td>
</tr>
<tr>
<td>int</td>
<td>32 bits</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>long</td>
<td>64 bits</td>
<td>-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807</td>
</tr>
<tr>
<td><strong>Real Numbers:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807</td>
</tr>
<tr>
<td><strong>Characters:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>char</td>
<td>16 bits unsigned</td>
<td>0 to 65535</td>
</tr>
<tr>
<td><strong>Booleans:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td>represents 1 bit</td>
<td>true or false (1 or 0)</td>
</tr>
<tr>
<td>Size is <em>undefined</em></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reference:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name of class</td>
<td>32 bits</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Notes: Mathematical operations

Order of operations: Please Excuse My Dear Aunt Sally: () ^ * / + -

Integer math: Huge rule: INT BY INT IS ALWAYS INT !!!

Example:  
5 + 2 = 7 // No big deal
8 – 3 = 5 // Again, no big deal
12 * 3 = 36 // Ditto.
80 / 4 = 20 // Ditto again.
9 / 2 = 4 // WHOA – Remember 3rd grade math!
9 % 2 = 1 // Modulus: returns the remainder

Double math: If you have one double in the equation, your answer is double. You must be careful though, because the above rule still applies.

5.0 + 2 = 7.0
5.0 + 2.0 = 7.0
8.0 – 3 = 5.0
12 * 3.0 = 36.0
80.0 / 4 = 20.0

Now the tricky part:
9 / 2.0 * 2 = 9.0 //Remember order of operations
9 / 2 * 2.0 = 8.0 //and the HUGE RULE
10 / 3 * 5.0 = 15.0
10 / 3.0 * 5 = 16.666666666666668

Character math: When you do math on characters, the compiler goes back to the ascii chart, using the values from there. Compare the code on the left to that on the right.

```
out.println('c' + 3); //prints 102
out.println('d' – 5); //prints 95
this even works:
out.println('c' * 'd'); //prints 9900
```

```
char x;
x = 'c' + 3;
out.println(x); //prints f
x = 'd' – 5'
out.println(x); //prints _
x = 'c' * 'd';
out.println(x); //prints ?
```

In class, we will only add /subtract integer values to / from characters.

String math: The only operation that works on strings is concatenation, which uses the + operator. Concatenation is simply “adding” one string to the end of the other.

```
String s = “Cat” + “Dog”;
String s1 = “Cat “ + “Dog”; //s1 now contains “Cat Dog”
String s2 = “Cat”;
    //s2 contains “Cat”
s2 = s2 + “armadillo”; //s2 now contains “Catarmadillo”
```

Gamel: CS1A and CS1
Assigning values:
You cannot assign a double number to an int. You will get an error that states, “Possible loss of precision.” You can assign an int to a double. The receiving value will convert the int to a double.

Examples:

```java
int x = 5, y;
double d = 7.0, z;

y = d; // double to int: SYNTAX ERROR
z = x; // z contains 5.0
```

Note that the computer assigns right to left. The right side of the equation is solved, and assigned to the value on the left. `x = x+1` certainly looks like a false statement. The computer however, calculates `x+1` and assigns the new value to `x`.

Shortcuts: The value of `x` changes for each of the following shortcuts.

Prefix `++`:
`++x;` // Same as `x = x + 1;`

Postfix `++`:
`x++;` // Same as `x = x + 1;`

Prefix `--`:
`--x;` // Same as `x = x – 1;`

Postfix `--`:
`x--;` // Same as `x = x – 1;`

So what is the difference between `x++` and `++x`? Not much. You can see the difference if you output these in `out.println(...);` For UIL, it’s important; for AP it’s not. 😊

Example:

```java
int x = 10;
out.println(x++); // outputs 10; then changes the value to 11;
vs.
int x = 10;
out.println(++x); // changes the value to 11; then outputs 11;
```

More Shortcuts:

```java
x+=8; // Same as `x = x + 8;`
x-=10; // Same as `x = x – 10;`
x/=5; // Same as `x = x / 5;`
x*=4; // Same as `x = x * 4;`
x%=3; // Same as `x = x % 3;`
```
Typecasting:

Often, you need a variable to be a different type from which it was declared. Since int by int is always int, you may need a double value to give you a double result. Considering student averages is the best example. If you have an 89.7 average, chances are you would like to have a 90 on your report card.

Consider the following:

```java
int total = 269;
int numgrades = 3;

double average = total / numgrades; //This result is 89!
```

If you typecast total OR numgrades, you can force it temporarily to double, for the sake of your equation.

```java
double average = (double) total / numgrades; // This result is 89.666666666666667
or
double average = total / (double) numgrades; // This result is 89.666666666666667
```

If total or numgrades had been defined as double, this would not be necessary. Often you NEED these values to be int and casting is then necessary.

In addition, the following does not cause an error:

```java
int x = (int) 3.14; //x would store 3
```
Assume you have the following variables and initial values.

```java
char ltr1 = 'c', ltr2 = 'C', ltr3;
int num1 = 20, num2 = 17, num3;
double d1 = 24.0, d2 = 30.50, d3;
String s1 = "Computer";
String s2 = "Science";
String s3;
```

Given the above declarations, let's look at how calculations are done in Java. Assume you have these initial values as each of the following statements is executed. Specify the calculated value.

1. `num2--;`  
   13. `num3 = 6 * 2 + 4.2;`
2. `++num1;`  
   14. `d3 = 6 * 2 + 4.2;`
3. `num2+=5;`  
   15. `num3 = 2 + 17 % 6;`
4. `ltr1+=2;`  
   16. `d3 = 2 + 17 % 6;`
5. `ltr2-=2;`  
   17. `num1/=4;`
6. `d3 = num1;`  
   18. `s3 = s1 + s2;`
7. `num3 = d2;`  
   19. `s3 = s1 + " " + s2;`
8. `d2*=3;`  
   20. `num2 *= 5;`
9. `d3 = (num1 + num2) / 2;`  
   21. `num1 /= 3;`
10. `d3 = (num1 + num2) / 2.0;`  
    22. `d3 = 25 / 4 * 3.0;`
11. `num3 = num1 % 7;`  
    23. `d3 = 25.0 / 4 * 3;`
12. `num3 = d1 % 7;`  
    24. `s3 = s2 + s1;`
1. int x = 5;
   x *= 4;
   x++;
   y = 3 + x;
   Final values: x: ______________________
y: ______________________

2. int number = 40;
   number /= 3;
   number *= 3;
   Final values: number: ______________________

3. double z = 10.0;
   int x = 5;
   z /= x;
   x++;
   z *= x;
   Final values: x: ______________________

4. String s1 = "Hex", s2 = "binary", s3 = "";
   s3 += s1;
   s2 += s1;
   s2 += s3;
   Final values: s1: ______________________
   s2: ______________________
   s3: ______________________

5. char letter = 'c';
   letter++;
   letter += 4;
   Final values: letter: ______________________

6. int number = 0;
   number *= 10;
   number += 4;
   number *= 10;
   Final values: number: ______________________

7. double salary = 40000;
   salary += 1500;
   salary = 40000;
   Final values: salary: ______________________

8. int change = 167;
   int q = 0, d = 0, n = 0, p = 0;
   Final values: change: ______________________
   q = change / 25;
   change % = 25;
   d = change / 10;
   change % = 10;
   n = change / 5;
   p = change % 5;
   Final values: q: ______________________
   d: ______________________
   n: ______________________
   p: ______________________
Mathematical operations Homework

Assume you have the following variables and initial values. Work each equation and get the new corresponding values. Assume that you re-initialize the values as stated for EACH equation.

char ltr1 = ‘P’, ltr2 = ‘C’;
int num1 = 20, num2 = 6, num3;
double d1 = 10.0, d2 = 3.5, d3;
String s1 = “Harry”;
String s2 = “Potter”;
String s3;

1. num1++; 17. d3 = d1 % 5.5;
2. num2--; 18. num3 = 4 * 5 + 6.2;
3. num2 += 5; 19. d3 = 4 * 5 + 6.2;
4. num1 -= 5; 20. d3 = (4 * 5 +11)/3;
5. ltr1 += 4; 21. d3 = (4 * 5 +11)/3.0;
6. ltr2 += 3; 22. d3 = (4.0 * 5 +11)/3;
7. ltr1 -= 3; 23. num3 = 4 + 13 % 4;
8. ltr2 -= 1; 24. d3 = 4 + 13 % 4;
9. d3 = num1; 25. d3 = 7.1 + 21/4;
10. num3 = d1; 26. num2 %= 2;
11. d3 = (num1 + num2 + 2) / 3; 27. num1 += 5;
12. d3 = (num1 + num2 + 2) / 3.0; 28. num2 -= 10;
13. d3 = (double)(num1 + num2 + 2) / 3; 29. num1 *= 3;
14. num3 = num1 % 3; 30. num1 /= 5;
15. num3 = num1 % 7; 31. s3 = s1 + s2;
16. num3 = d1 % 5; 32. s3 = s2 + s1;
33. s3 = s1 + “ “ + s2;
Answer Key: Mathematical Operations HW

1. 21
2. 5
3. 11
4. 15
5. T
6. F
7. M
8. B
9. 20.0
10. Syntax Error
11. 9.0
12. 9.33
13. 9.33
14. 2
15. 6
16. Syntax Error
17. 4.5
18. Syntax Error
19. 26.2
20. 10.0
21. 10.33
22. 10.33
23. 5
24. 5.0
25. 12.1
26. 0
27. 25
28. -4
29. 60
30. 4
31. HarryPotter
32. PotterHarry
33. Harry Potter
Math Operations Program Assignments

**Program Minutes:**

Save as minutes.java

Write a Java program to calculate the number of minutes in one year. There is no input for this program. You can assign values directly in your program, or you can do all of your calculations in one statement. You may not use a calculator and then out.print your answer. Your program must calculate the answer.

Example:

```java
int min_per_hour = 60; //or int totalMinutes = ...;
```

**Program Cans**

Save as cans.java

Write a Java program to solve the problem given below.

The number of gallons of soda made in the U. S. in 2002 was 4,894,000,000. (You must use double, because this number is larger than an int or even a long can store.) A 12 oz. Can is 4.75 inches tall. Suppose all the soda made in 2002 was placed in 12 oz. cans and that these cans were then stacked one on top of the other. Write a program to compute the following quantities:

a. The number of cans in the stack.
b. The height of the stack in inches.
c. The height of the stack in miles.
d. The number of times the stack would reach the moon.

Useful facts:

a. There are 128 oz. In a gallon.
b. There are 5280 feet in a mile.
c. The distance to the moon is 239,000 miles.

Keep in mind, there is no input for this program. You can assign values directly in your program, but you don't need variables to hold ounces in a gallon or inches in a foot. (These values are always 128 and 12, respectively, so you can use the constants for your calculations.) I recommend to use variables for a – d above.

Example:

```java
double gallons = 4894000000.0; //note no commas! (It is not String.)
   // The .0 is required because this is such a large number, it will not fit into 4 bytes
   // for an int; it must be double (right AND left side).
```
**Input: Scanner Class**

The Scanner class is used to input data in java programs. We will study input more in our next unit. For now, we will only input numbers. To use Scanner, you must import:

```java
import java.util.*; or import java.util.Scanner;
```

```java
Scanner sc = new Scanner(System.in);  //declaration of Scanner object
int x = sc.nextInt();  //reads an integer (int)
double y = sc.nextDouble();  //read a double number (double)
boolean b = sc.nextBoolean();  //reads a boolean value
String s = sc.next();  // reads to white space  //reads one word in String
String l = sc.nextLine();  // reads to \n  //reads entire String, including spaces
```

Be sure to include a prompt when reading data, otherwise the user will not know what to enter when the computer is on a blank screen.

Example:

```java
out.print("Enter grade: ");
grade = sc.nextInt();

out.print("Enter salary: ");
salary = sc.nextDouble();

out.print("Enter name: ");
fullName = sc.nextLine();

out.print("Enter an integer: ");
x = sc.nextInt();
```
Math Operations w/ Input Programs

Program Time TO SECONDS:  
Write a program that reads values representing a time in hours, minutes and seconds. Then print the same time in seconds. For example, 1 hour, 28 minutes and 42 seconds is equal to 5322 seconds.)

Sample:

Enter hours: 1
Enter minutes: 28
Enter seconds: 42

Total: 5322

Modulus Programs

Program Seconds TO TIME:
Create a new version of Time To Seconds, that works in reverse. That is, read a value representing a number of seconds, then print the same amount of time in hours, minutes and seconds. (For example, 9999 seconds is equal to 2 hours, 46 minutes, and 39 seconds.)

Sample:

Enter seconds: 9999
Hours: 2
Minutes: 46
Seconds: 39

Program Separate:
Write a program that prompts the user to enter a 3-digit integer. Your program should separate the digits and output 1st, 2nd and 3rd digits. Example execution:

Sample:

Enter a 3-digit integer: 149

1st digit: 1
2nd digit: 4
3rd digit: 9

Program Change:
Write a program that prompts the user to enter the total amount of change in cents. Your program should calculate the number of quarters, dimes, nickels and pennies in the given amount. The following is an example execution:

Sample:

Enter total amount of change in cents: 168
Quarters: 6
Dimes: 1
Nickels: 1
Pennies: 3
### CS1AP: TEST 2 B
### Java Basics through Math Operations

Choose one answer from the right column to match the descriptions in the left column. Answers on right will appear once on the left, or not at all.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>_____</td>
<td>Escape sequence to print backslash</td>
<td>A. \n</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>_____</td>
<td>Terminates Java statements</td>
<td>B. character</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>_____</td>
<td>Mark the beginning and end of a Java program</td>
<td>C. char</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>_____</td>
<td>Library imported to use Scanner (input)</td>
<td>D. int</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>_____</td>
<td>Escape sequence to go to the next tab stop</td>
<td>E. \t</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>_____</td>
<td>Escape sequence to force a new line</td>
<td>F. String</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>_____</td>
<td>Data type to hold a '?'</td>
<td>G. \</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>_____</td>
<td>Data type to hold 17</td>
<td>H. /* */</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>_____</td>
<td>Data type to hold 17.25</td>
<td>I. static java.lang.System.*;</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>_____</td>
<td>Data type to hold &quot;KHS&quot;</td>
<td>J. { } (curly brackets)</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>_____</td>
<td>Data type to hold the values true or false</td>
<td>K. [ ] (square brackets)</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>_____</td>
<td>Library imported to use out.println(); instead of System.out.println()</td>
<td>L. ; (semicolon)</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>_____</td>
<td>Symbol(s) used for a one-line comment statement</td>
<td>M. println</td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>_____</td>
<td>Symbol(s) used for a block comment statement</td>
<td>N. double</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>_____</td>
<td>Assuming the rest of the program is coded correctly, what is the output of the following program segment?</td>
<td>O. //</td>
<td></td>
</tr>
</tbody>
</table>

```java
out.print "The quick brown fox";
out.print "jumps over the lazy dog."
```

A. The quick brown fox jumps over the lazy dog.
B. The quick brown foxjumps over the lazy dog.
C. The quick brown fox jumps over the lazy dog.
D. Error message
16. Assuming the rest of the program is coded correctly, what is the output of the following program segment?
   ```java
   out.println("The quick brown fox");
   out.println("jumps over the lazy dog.");
   ```
   A. The quick brown fox jumps over the lazy dog.
   B. The quick brown fox jumps over the lazy dog.
   C. The quick brown fox
      jumps over the lazy dog.
   D. Error message

17. Assuming the rest of the program is coded correctly, what is the output of the following program segment?
   ```java
   out.println("The quick brown fox");
   out.println("jumps over the lazy dog.");
   ```
   A. The quick brown fox jumps over the lazy dog.
   B. The quick brown fox jumps over the lazy dog.
   C. The quick brown fox
      jumps over the lazy dog.
   D. Error message

18. Assuming the rest of the program is coded correctly, what is the output of the following program segment?
   ```java
   int age = 25;
   String firstName = "John";
   String lastName = "Doe";
   out.println(firstName + " " + lastName + " " + age);
   ```
   A. John Doe 25
   B. John Doe 25
   C. John
      Doe
      25
   D. Error Message

19. Assuming the rest of the program is coded correctly, what is the output of the following program segment?
   ```java
   int age = 25;
   String firstName = "John";
   String lastName = "Doe";
   out.println(firstName + lastName + age);
   ```
   A. John Doe 25
   B. John Doe 25
   C. John
      Doe
      25
   D. Error Message
20. **Segment A**
   ```java
   int a = 4.5;
   int b = 7.2;
   out.println(a);
   out.println(b);
   ```
   **Segment B**
   ```java
   int a = 100;
   int b = 200;
   out.println(a);
   out.println(b);
   ```

   Assuming the rest of the program is coded correctly, what is true about the comparison of these two segments?
   A. Segment A and segment B are both correct.
   B. Segment A and segment B are both incorrect.
   C. Segment A is correct and segment B is not correct.
   D. Segment A is incorrect and segment B is correct.

21. The expression `(double) 65` yields: //Ascii 65 is ‘A’
   A. 65 //Ascii 97 is ‘a’
   B. 65.0
   C. A
   D. a

22. The expression `(char) 65` yields: //Ascii 65 is ‘A’
   A. 65 //Ascii 97 is ‘a’
   B. 65.0
   C. A
   D. a

23. What is the output of this command?: `out.println("1 + 2 = " + 1 + 2);`
   A. 3 = 3
   B. 1 + 2 = 3
   C. 1 + 2 = 12
   D. 3 = 12

24. What is the output of this command?: `out.println("1 + 2 = " + (1 + 2));`
   A. 3 = 3
   B. 1 + 2 = 3
   C. 1 + 2 = 12
   D. 3 = 12

25. Which of the following is NOT a primitive data type in Java?
   A. int
   B. double
   C. String
   D. char

   Write V for Valid or I for Invalid beside each of the following variable names:

26. __________ Employee Address
    30. __________ main
27. __________ String
    31. __________ Number_1
28. __________ EmployeeAddress
    32. __________ 1Number
29. __________ StAte
    33. __________ Remainder%
Short Answer: Write the correct answer for each question.

34. Our HUGE rule is:
_________________ by ________________ is always ________________!

35. Write one or more commands to output *Katy Tigers* on the first line and *State Champs* on the second line.

36. Write the statement to declare the integer variables *side1*, *side2* and *area*.

37. Using the variables from #36, write the statement to calculate the area of the rectangle and assign the results the variable *area*.

38. Write the statement to output your first and last name separated by a tab.

39. Write the statement to output *Sum:* followed by the value stored in the variable *sum*.

40. Write the statement to properly declare variables for *gpa*, hourly wages and tax rate.

41. Write a comment statement that says *Java is Fun!*

Given the following declarations, write the correct answer assuming each statement is executed immediately after the declarations. Be sure to show a decimal value when necessary. If the statement will not compile, write Error.

class  | ltr1 = ’i’, ltr2 = ’I’, char ltr3;  
int num1 = 60, num2 = 29, num3;  
char ltr1 = ’i’, ltr2 = ’I’, char ltr3;  
int num1 = 60, num2 = 29, num3;  

42. num1++;  
43. ltr1--;  
44. num2+=5;  
45. num1-=10;  
46. ltr2+=5;  
47. d3 = num1;  
48. num3 = d1;  
49. d3 = (num1 + num2) / 2;  
50. d3 = (double)(num1 + num2) / 2;  
51. num3 = num1 % 8;  
52. num3 = d1 % 5;  
53. d3= d1 % 3.5;  
54. d3 = (7 * 3 )/4;  
55. d3 = (7 * 3)/4.0;  
56. num3 = 4 + 13 % 3;  
57. num3 = (21 / 4 * 2);  
58. num1/=10;  
59. s3 = s1 + s2;
60. How many bits in the following data types?

- char
- double
- short
- long
- int
- float
- boolean
- byte
- reference (address)

(61 - 66) There are six errors in the following program. Correct each error.
DO NOT JUST CIRCLE IT – FIX IT!

```java
import static java.lang.System.*;
import java.util.*;

public class errors {
    public static void main(String[] thisisavalidname) {
        // FIND SIX ERRORS IN THIS PROGRAM.
        Scanner sc = new Scanner(System.in);
        int grade1, grade2, grade3
        double average;
        out.print("Enter first grade: ");
        grade1 = sc.nextInt();
        out.print("Enter second grade: ");
        grade2 = sc.nextInt();
        out.print("Enter third grade: ");
        grade3 = sc.nextInt();

        total = grade1 + grade2 + grade3;
        average = total / 3;

        out.println("The average is ", average);
        out.println();
    }
}
```
import static java.lang.System.*;  
import java.util.*;  
public class errors  
{  
    public static void main(String[] thisisavalidname)  
    {  
        \FIND SIX ERRORS IN THIS PROGRAM.  
        Scanner sc = new Scanner(System.in);  
        int grade1, grade2, grade3, total;  
        double average;  
        out.print("Enter first grade: ");  
        grade1 = sc.nextInt();  
        out.print("Enter second grade: ");  
        grade2 = sc.nextInt();  
        out.print("Enter third grade: ");  
        grade3 = sc.nextInt();  
        total = grade1 + grade2 + grade3;  
        average = total / 3.0;  
        out.println("The average is "+ average);  
        out.println();  
    }  
}
Storing Various Base Values into an Integer

Storing a Binary Value into an Integer

Format:

```java
int name = 0bbinaryNumber;
```

Example:

```java
int b = 0b011;
System.out.println(b);
```

Example Output:

3

Storing an Octal Value into an Integer

Format:

```java
int name = 0octalNumber;
```

Example:

```java
int o = 011;
System.out.println(o);
```

Example Output:

9

Storing a Hexadecimal Value into an Integer

Format:

```java
int name = 0xHexadecimalNumber;
```

Example:

```java
int h = 0xA1;
System.out.println(h);
```

Example Output:

161

Getting Various Base Values from an int

Format:

```java
String name = Integer.toString(value, base);
```

Example:

```java
String b = Integer.toString(50,2);
String o = Integer.toString(50,8);
String h = Integer.toString(50,16);
System.out.println(b);
System.out.println(o);
System.out.println(h);
```

Example Output:

110010
62
32