1) Write a program that does the following: Create seven variables, one for each of the primitive number types in Java, and initialize each variable with any appropriate value. Print out the name of each variable and its value. Modify the value of each variable with an assignment statement and print out the names of the variables and their new values. Put your code here:

|  |
| --- |
| boolean \_bool = true;  short \_short = 2;  int \_int = 3;  long \_long = 4;  float \_float = 5;  double \_double = 6;  byte \_byte = 7;  System.out.println("\nIntialization\nBool:\t" + \_bool);  System.out.println("Short:\t" + \_short);  System.out.println("Int:\t" + \_int);  System.out.println("Long:\t" + \_long);  System.out.println("Float:\t" + \_float);  System.out.println("Double:\t" + \_double);  System.out.println("Byte:\t" + \_byte);  \_bool = false;  \_short = 101;  \_int = 102;  \_long = 103;  \_float = 104;  \_double = 105;  \_byte = 106;  System.out.println("\nAfter Assignment\nBool:\t" + \_bool);  System.out.println("Short:\t" + \_short);  System.out.println("Int:\t" + \_int);  System.out.println("Long:\t" + \_long);  System.out.println("Float:\t" + \_float);  System.out.println("Double:\t" + \_double);  System.out.println("Byte:\t" + \_byte); |

Next, create seven constants, one for each of the primitive number types in Java. Print the name of the constant and its value. Put your code here:

|  |
| --- |
| final boolean const\_bool = true;  final short const\_short = 71;  final int const\_int = 72;  final long const\_long = 73;  final float const\_float = 74;  final double const\_double = 75;  final byte const\_byte = 76;  System.out.println("\nConstants\nBool:\t" + const\_bool);  System.out.println("Short:\t" + const\_short);  System.out.println("Int:\t" + const\_int);  System.out.println("Long:\t" + const\_long);  System.out.println("Float:\t" + const\_float);  System.out.println("Double:\t" + const\_double);  System.out.println("Byte:\t" + const\_byte); |

What happens if you try to assign a value to a constant?

**Cannot assign a value to final variable ‘*variable name’,* is what the IDE reports. In any language the compiler will throw an error when trying to change the “constant” variables.**

2) Execute the program below. Each invocation of println outputs an arithmetic expression. The first two println commands are followed by comments that describe the operations that occur in each expression. Complete the program by adding a comment after each println statement that describes all the arithmetic operations that occur when evaluating the expression that is printed.

|  |
| --- |
| public class Expressions  {  public static void main(String[] args)  {  int a = 3;  int b = 4;  int c = 5;  int d = 17;  System.out.println((a + b)/ c);  // 3 and 4 are added with sum 7  // 7 is divided by 5 with quotient 1  System.out.println(a + b / c);  // 4 is divided by 5 with quotient 0  // 3 is added to 0 with sum 3  System.out.println(a++);  **// Displays variable 'a' at 3, then does**  **// post-increment, changing value to 4**  System.out.println(a--);  **// Displays variable 'a' at 4, then does**  **// post-decrement, changing value to 3**  System.out.println(a + 1);  **// 1 and 3 added with sum 4**  System.out.println(d % c);  **// 17 modulus 5, displaying remainder of 2**  System.out.println(d / c);  **// Integer division, 17 divided by 5 is 3**  System.out.println(d % b);  **// 17 modulus 4, displaying remainder of 1**  System.out.println(d / b);  **// Integer division, 17 divded by 4 is 6**  System.out.println(d + a / d + b);  **// 3 divided by 17 = 0 (int division)**  **// 17 is added to 4 with sum 21**  System.out.println((d + a) / (d + b));  **// 17 is added to 3 with sum 20**  **// 17 is added to 4 with sum 21**  **// 20 is divided by 21, integer division equals 0**  System.out.println(Math.sqrt(b));  **// Square root of 4 is 2**  **// math.sqrt automatically casts double on**  **// argument. Causing output to be 2.0**  System.out.println(Math.pow(a, b));  **// 3 to the power of 4, product of 81**  System.out.println(Math.abs(-a));  **// Absolute value of -3 is 3**  System.out.println(Math.max(a, b));  **// Compares the arguments, which are 3 and 4**  **// displays the maximum, 4**  }  } |

3) Write a program that prompts the user to enter two integers. Print the smaller of the two numbers entered. You’ll need to use a Scanner and a Math method. Put your code here:

|  |
| --- |
| import java.util.Scanner;  Scanner input = new Scanner(System.in);  int one, two;    System.out.print("Please enter a first number: ");  one = input.nextInt();  System.out.print("Please enter a second number: ");  two = input.nextInt();    System.out.println(Math.min(one, two)); |

4) Suppose you have 5 1/2 gallons of milk and want to store them in milk jars that can hold up to 0.75 gallons each. You want to know ahead of time how many completely filled jars you will have. The following program has been written for that purpose. What is wrong with it? Why? How can you fix it?

**By having the variable completelyFilledJars be an *int* and then assigning it by using *double* division causes compiler errors. However if it was reversed, *Double* being assigned by *Int* division would be okay. To fix it simply change the *int* to *double***

|  |
| --- |
| public class MilkJarCalculator  {  public static void main(String[] args)  {  double milk = 5.5; // gallons  double jarCapacity = 0.75; // gallons  int completelyFilledJars = milk / jarCapacity;    System.out.println(completelyFilledJars);  }  } |

5.1) You want to know how many feet are in 3.5 yards, and how many inches are in 3.5 yards. You write the following program for that purpose:

|  |
| --- |
| public class DistanceConverter  {  public static void main(String[] args)  {  double yards = 3.5;  double feet = yards \* 3;  double inches = feet \* 12;    System.out.println(yards + "yards are" + feet + "feet");  System.out.println(yards + "yards are" + inches + "inches");  }  } |

The problem with the program above is that using "magic numbers" makes it hard to maintain and debug. Modify the program so that it uses constants to improve legibility and make it easier to maintain. Put your code here:

|  |
| --- |
| final double INCHES\_IN\_A\_FOOT = 12;  final double FEET\_IN\_A\_YARD = 3;  final double DISTANCE\_FOR\_CONVERSION = 3.5;  double yards = DISTANCE\_FOR\_CONVERSION;  double feet = yards \* FEET\_IN\_A\_YARD;  double inches = feet \* INCHES\_IN\_A\_FOOT;  System.out.println(yards + "yards are" + feet + "feet");  System.out.println(yards + "yards are" + inches + "inches"); |

5.2) Run the DistanceConverter program from 5.1.) What is the output? What change(s) would you make to the program to make the output more readable? Put your code here:

|  |
| --- |
| **OUTPUT OF ‘YOUR’ CODE:**  **3.5yards are10.5feet**  **3.5yards are126.0inches** |

**The changes I would make to the code to make it more readable would be to output into a table for easier readability. However this is just my preference as there is no ‘absolutely correct’ way to make output easier to understand;**

|  |
| --- |
| **MY CHANGE TO THE CODE:**  System.out.println(yards + " yards is:\n" + feet + "\tfeet\n" + inches + "\tinches");  **OUTPUT OF MY CODE:**  **3.5 yards is:**  **10.5 feet**  **126.0 inches** |

6) Adding (incrementing) or subtracting (decrementing) the value one from an integer variable is a common, everyday operation. To increment an int variable x, we could code

x = x + 1;

As an alternative, we could use the special operators ++ and -- to increment and decrement a variable. Use the first method to increment x in the program below. Print the value of x after incrementing. Use the ++ operator to increment y in the program below. Print the value of y after incrementing.

|  |
| --- |
| public class IncrementDemo  {  public static void main(String[] args)  {  int x = 10;  int y = -3;  // MY CODE //  x = x + 1;  y++;  System.out.println("Results:\nx:\t" + x + "\ny:\t" + y);  // END MY CODE //  }  }  **OUTPUT:**  **Results:**  **x: 11**  **y -2** |

7.1) An *annuity* (sometimes called a reverse mortgage) is an account that yields a fixed payment every year until it is depleted. The present value of the annuity is the amount that you would need to invest at a given interest rate so that the payments can be made.

The present value of an annuity (PVann) at the time of the first deposit can be calculated using the following formula:

PVann = PMT · ({[(1 + i)n - 1 - 1] / i } / (1 + i)n - 1 + 1)

where:

PMT: periodic payment

i: periodic interest or compound rate

n: number of payments

What is the present value of an annuity that will pay out $10,000 in each of the next 20 years if the interest rate is 8 percent?

Write a program to calculate the present value of an annuity for these values. Remember that you can use Math.pow(x, y) to calculate *xy*.

What is your program? Put your code here:

|  |
| --- |
| double period\_payment = 10000;  double periodic\_interest = .08;  double number\_of\_payments = 20;  double present\_annuity;  //USING THESE TWO TO SIMPLIFY EQUATION  double formula\_numerator = (Math.pow((1+periodic\_interest), (number\_of\_payments-1))-1)/periodic\_interest;  double formula\_denominator = (Math.pow((1+periodic\_interest), (number\_of\_payments-1))+1);  present\_annuity = period\_payment\*(formula\_numerator/formula\_denominator);  System.out.println("Present Value: \t$" + Math.round(present\_annuity));  **OUTPUT:**  **Present Value: $77970** |

7.2) Modify the program you created in 7.1) so that the user can provide the values for pmt, i, and n through the console. Put your code here:

|  |
| --- |
| Scanner input = new Scanner(System.in);    double period\_payment;  double periodic\_interest;  double number\_of\_payments;  System.out.print("Please enter the periodic payment: ");  period\_payment = input.nextDouble();  System.out.print("Please enter the periodic interest: ");  periodic\_interest = input.nextDouble()/100;  System.out.print("Please enter the number of payments: ");  number\_of\_payments = input.nextDouble();  //USING THESE TWO TO SIMPLIFY EQUATION  double formula\_numerator = (Math.pow((1+periodic\_interest), (number\_of\_payments-1))-1)/periodic\_interest;  double formula\_denominator = (Math.pow((1+periodic\_interest), (number\_of\_payments-1))+1);  double present\_annuity = period\_payment\*(formula\_numerator/formula\_denominator);  System.out.println("Present Value: \t$" + Math.round(present\_annuity));  **OUTPUT (WITH MY INPUT):**  **Please enter the periodic payment: 10000**  **Please enter the periodic interest: 8**  **Please enter the number of payments: 20**  **Present Value: $77970** |

8.1) What is the output of the following program? Why?

public class AverageCalculator

{

public static void main(String[] args)

{

int age1 = 18;

int age2 = 35;

int age3 = 50;

int age4 = 44;

double averageAge = (age1 + age2 + age3 + age4) / 4;

System.out.println(averageAge);

}

}

**The ouput is 36, the answer is supposed to be 36.75 however with integer division (integer truncation) it rounds off the remainder;**

8.2) Fix the program in 8.1) so that it yields the correct result. Put your code here:

|  |
| --- |
| int age1 = 18;  int age2 = 35;  int age3 = 50;  int age4 = 44;  double averageAge = (double)(age1 + age2 + age3 + age4) / 4;  System.out.println(averageAge);  **OUTPUT:**  **36.75** |

9.1) What is the output of the following program? Why?

public class PercentagePrinter

{

public static void main(String[] args)

{

double probability = 8.70;

int percentage = (int) (100 \* probability);

System.out.println(percentage);

}

}

**The output of the program is 869, the actual answer would be 870, but doing arithmetic with *int’s* is a not a good habit to use. Integer Truncation is the main reason for the problem, whenever something is typecast, or is itself an int, you will experience these problems as an integer cannot represent *real numbers,* meaning numbers with digits after the decimal.**

9.2) Fix the program from 9.1) so that it displays the correct result. Remember that you can use Math.round to convert a floating-point value to its *closest* integer. Put your code here:

|  |
| --- |
| double probability = 8.70;  double percentage = Math.round(100 \* probability);  System.out.println(percentage); |

**It should be noted that this simply making the *percentage* variable a double you’ll experience “round-off errors” which is caused by computer arithmetic which only shows approximations of numbers and not ‘real numbers’**

10) Using the substring method and concatenation, write a sequence of commands that will extract characters from inputString = "The quick brown fox jumps over the lazy dog" to make outputString = "Tempus fugit". Then print outputString. Put your code here:

|  |
| --- |
| **String inputString = "The quick brown fox jumps over the lazy dog";**    **String outputString; //Should become: Tempus fugit**    **String word\_Tempus = inputString.substring(0,1) + //T**  **inputString.substring(2,3) + //E**  **inputString.substring(22,24) + //MP**  **inputString.substring(5,6) + //U**  **inputString.substring(24,25)+ //S**  **inputString.substring(3,4); //white\_space**    **String word\_Fugit = inputString.substring(16,17) + //F**  **inputString.substring(5,6) + //U**  **inputString.substring(42) + //G**  **inputString.substring(6,7) + //I**  **inputString.substring(31, 32); //T**    **outputString = word\_Tempus + word\_Fugit;**    **System.out.println(outputString);**  **OUTPUT:**  **Tempus fugit** |