**CMSC 155 Spring 2020**

**Homework 4 (135 points available)**

**DUE: Midnight Thursday Feb 13**

**Always hand in your code PLUS how you tested it and the results obtained.**

1. Do Exercise 4. (20 points)
2. What happens if: (10 points)
3. An instance variable is not declared as public OR private?
4. An object is printed and there is no toString method in its class?

A bunch of letters and numbers will appear that often give the location of file.

1. A class has no constructor?

If a class has no constructor than the variables are not initialized for the current methods.

1. A static method tries to access an instance variable?
2. A non-static method tries to access a static variable?

1. Overloading

Assume the following function is defined: public static int func(int x, double y)

Which of the following function definitions **would be valid in the same class**? For those that are not valid, EXPLAIN why. (10 points)

* 1. public static int func(int x, int y, double z)
  2. public static int func(int x, String y)
  3. public static double func(int x, double y)
  4. public static int func(int a, double b)
  5. public static int func(double x, int y)

1. Modify your test code so that you create another student where the studentID is set automatically (is not passed to the constructor). Leave the original code that created a student object as it was. (5 points)

package Week5;  
  
public class StudentCourseTest {  
  
 public static void main(String[] args) {  
 Student stu1 = new Student("Josh", 500);  
 Course course1 = new Course("CMSC",155, 4);  
 System.*out*.println(stu1);  
 Student stu2 = new Student("Justin");  
 System.*out*.println(stu2);  
 Student stu3 = new Student("Jordan");  
 System.*out*.println(stu3);  
 }  
}

Josh 500

Justin 100

Jordan 101

1. Modify the Student class by creating a static variable called nextID. Create a second version the constructor to use the static variable to set the student id and then increment the static variable. (10 points)

package Week5;  
  
public class Student {  
  
 public static int *nextId* = 100;  
  
 private String name;  
 private int id;  
  
  
 public Student(String name, int id){  
 this.id=id;  
 this.name=name;  
 }  
  
 public Student(String name){  
 this(name, *nextId*++);  
 }  
  
 public void changeName(String name){  
 String newName = "Jared";  
 name = newName;  
 }  
  
 public String toString() {  
 return name + " " + id;  
 }  
}

1. Add and test a method in the Student class that returns true or false depending on whether the first student’s ID is numerically smaller than the second’s. (10 points)

E.g. Student bob1 = new Student("Bob Smith", 971);

Student bob2 = new Student("Bob Jones", 4321);

bob1.precedesById(bob2) should return true

bob2.precedesById(bob1) should return false

1. Assume the Course class has a method

public Course doSomething(Course course)

Assume a non-static method in the Course class has a local variable myCourse of type Course. Which of the following is validin that method? If invalid, explain why.

(15 points)

* 1. doSomething(myCourse);

* 1. System.*out*.println(doSomething(myCourse));

* 1. doSomething(Course);

* 1. myCourse.doSomething(myCourse);

* 1. myCourse = myCourse.doSomething(myCourse);

* 1. course = myCourse.doSomething(myCourse);

* 1. Course course = myCourse.doSomething(myCourse);

1. Write a test code to a voting machine object. The test code should instantiate a voting machine object with an issue name. Then it should exercise the voteYes() method (cast a single yes vote) twice and the voteNo() method once. Print the object, then clear it and print it again. See the next problem for more details. (5 points)

package Week5;  
  
public class VotingMachineObjects {  
  
 public static void main(String[] args) {  
 VotingMachine obj1 = new VotingMachine(0, 0, "President");  
 obj1.voteYes();  
 obj1.voteYes();  
 obj1.voteNo();  
 System.*out*.println(obj1);  
 obj1.clearMachine();  
 System.*out*.println(obj1);  
 }  
}

Number of YES = 2 Number of NO = 1

Number of YES = 0 Number of NO = 0

1. Write a VotingMachine class. The class should have the following instance variables: numberYesVotes (an int) and numberNoVotes (an int) and a String with the name of the issue being voted on. It should have the following five methods in addition to a constructor and a toString method:
   1. voteYes that takes no parameters and does not return anything (void return type) but increments numberYesVotes
   2. voteNo that takes no parameters and does not return anything but increments numberNoVotes
   3. returnYesCount which takes no paramters. It returns numberYesVotes.
   4. returnNoCount which takes no parameters. It returns numberNoVotes.
   5. clearMachine which takes no parameters and does not return anything. It should reset numberYesVotes and numberNoVotes to zero.
   6. The toString method should call the returnYesCount() and returnNoCount() methods.

(15 points)

package Week5;  
  
public class VotingMachine {  
  
 private int numberYesVotes;  
 private int numberNoVotes;  
 private String issue;  
  
 public VotingMachine(int numberYesVotes, int numberNoVotes, String issue){  
 this.numberYesVotes = numberYesVotes;  
 this.numberNoVotes = numberNoVotes;  
 this.issue = issue;  
 }  
  
 public void voteYes(){  
 numberYesVotes++;  
 }  
  
 public void voteNo(){  
 numberNoVotes++;  
 }  
  
 public int returnYesCount(){  
 return numberYesVotes;  
 }  
  
 public int returnNoCount(){  
 return numberNoVotes;  
 }  
  
 public void clearMachine(){  
 numberNoVotes = 0;  
 numberYesVotes = 0;  
 }  
  
 public String toString(){  
 return "Number of YES = " + returnYesCount() + " Number of NO = " + returnNoCount();  
 }  
  
}

1. **Explain:** (10 points)

1. The difference between **instance variables and local variables**.

Instance variables can be seen/used by all methods in a class where as local variables can only be used by the method they are initiated in.

1. The principle of encapsulation. Give some examples of how encapsulation can be violated.

Encapsulation binds together code in a class and makes it act as a single thing. Encapsulation can be violated if there are not getter and setters within the code.

1. Why it is good programming practice to call one constructor from another, instead of having each constructor initialize the instance variables.

Because it would just be repetitious if you initialize the same instance variables in multiple constructors.

1. The use of the variable this inside a class.

*this* is used to classify a current object as an action.

1. Why it is a good idea to put a toString method in a class.

It is a good idea because then the code that is returned to print, will not look like a bunch of scrambled letters and numbers.

1. Create an ExtraMathFunctions class that will perform mathematical functions that are not in the general Math library. Specifically write three **stati**c methods (including the main method) as follows: (10 points)
   1. The first method should **return** what a given percentage of a number is. It should two parameters, an integer that represents the percentage the user wishes to find and a double, the number from which to calculate the percentage. For example, if you call this method percentOf, the call percentOf(20, 115) should return 23.
   2. The second method should take two doubles as parameters and return a double that is the percentage the first number is of the second. For example, if this method is called findPercent, the call findPercent(100, 200) should return 50.
   3. Test these two method by calling them from a main method.

package Week5;  
  
public class ExtraMathFunctions {  
  
 public static void main(String[] args) {  
 ExtraMathFunctions percent1 = new ExtraMathFunctions();  
 System.*out*.println(*percentOf*(20,115));  
 ExtraMathFunctions percent2 = new ExtraMathFunctions();  
 System.*out*.println(*findPercent*(100,200));  
 }  
  
 public static double percentOf(int percentage, double calculatePercent){  
 return (percentage \* .01) \* calculatePercent;  
 }  
  
 public static double findPercent(double num1, double num2){  
 return (num1 / num2) \* 100;  
 }  
}

23.0

50.0

1. Why are these methods good candidates to be a static methods? (5 points)

These methods are good candidates for static methods because instead of creating instances for each method we can use the class wide instances for each static method rather than inefficient non static methods each needing their own local variables.

1. **Difference between & and &&**.

In Java, **&&** denotes the boolean operator "And".

However, Java also has an operator **&**.

Sometimes & and && produce the same result, but sometimes they do not. Compare the results of the following:

* 1. (1 < 2) & (2 < 3)🡪 true (1 < 2) && (2 < 3)🡪 true
  2. 1 & 2 🡪 0 1 && 2 🡪 This is an error

What is & actually computing? (This will require some research.) (10 points)

& is actually computing the same thing as && but doing it differently. & compares both of the given statements and checks if they are true. If the first one is false, it will stick check if the second one is true whereas && will only check the second statement if the first is true. && is more efficient in that way.

There is a similar difference between **|** and **||**.

Always use && for "And" and **||** for "Or".