University of Maryland College Park

Dept of Computer Science

CMSC132

Midterm I

Last Name (PRINT): ____

First Name (PRINT): ____

University Directory ID (e.g., umcpturtle) ________________________________

Lab TA (Circle One):

0101/0102 Medhi	**0201Jeremy**	**0203/0204 Stefani**	**0303/0304 Raghav**	**0402 Solomon**	**0404 Victor**
0103/0104 Nisarg	**0202 William**	**0301/0302 Meena**	**0401 Avery**	**0403 Jihoon**	**Honors**

I pledge on my honor that I have not given or received any unauthorized assistance on this examination.

Your signature: ____

Instructions

- This exam is a closed-book and closed-notes exam.
- Total point value is 200 points.
- The exam is a 50 minutes exam.
- Please use a pencil to complete the exam.
- WRITE NEATLY.
- **Your code must be efficient.**
- **You don't need to use meaningful variable names; however, we expect good indentation.**

Grader Use Only

#1	Problem #1 (Miscellaneous)	38	
#2	Problem #2 (Class Implementation)	162	
#3	Honors	12	
Total	Total	200/212	

Problem #1 (Miscellaneous)

1. (3 pts) Does the following class rely on encapsulation? If it does, write YES; otherwise NO and fix it. Notice the getName method returns a string and not a StringBuffer.

```
public class Name {
     StringBuffer name;

     public String getName() {
            return name.toString();
     }
}
```

2. (3 pts) What will take place when the following code fragment is executed?

```
            Object lc = null;
            boolean val = lc instanceof Object;
```

   a. An exception will be generated.
   b. false will be assigned to val.
   c. true will be assigned to val.
   d. None of the above.

3. (7 pts) A class called **Hero** is defined as follows:

```
public class Hero {
     private int power;

     public int increasePower(double delta) {
            return power += (int) delta;
     }
}
```

Another class called **SuperHero** extends **Hero** and can define one of the methods below. Indicate what will happen if one of the methods below were to be added to the **SuperHero** class. Use RIDE to indicate the method will override increasePower, LOAD to indicate it will overload increasePower, and ERROR if it will generate a compilation error.

   a. `public int increasePower(double delta) { return 4 + (int)delta; }`
   b. `protected int increasePower(double delta) { return 4 + (int)delta; }`
   c. `int increasePower(double delta) { return 4 + (int)delta; }`
   d. `public double increasePower(double delta) { return 4.0 + delta; }`
   e. `public int increasePower() { return power += 100; }`
   f. `public double increasePower() { return power += 100.00; }`
   g. `public increasePower() { return power += 100.00; }`

4. (3 pts) Would the **Hero** class have a default constructor if the following method where added to the class? **Yes**/**No**.

```
public Hero(int powerIn) { power = powerIn; }
```

5. (3 pts) Assume the **Hero** and **SuperHero** classes above define a method with the following signature:

```
public void origin()
```

If Java were using dynamic binding, which method (the one in **Hero** or in **SuperHero**) will be used when the following code is executed?

```
Hero h = new SuperHero();
h.origin();
```

6. (3 pts) For the code below, either write what value for x will printout or, if it will not compile, write **will not compile**.

```
public class Exam1 {
      int x = 5;

      public Exam1() { x = 6; }

      { x = 7; }
      public static void main(String[] args) {
             System.out.println(new Exam1().x);
      }
}
```

7. (3 pts) You are given two classes A and B without no information about them except they work together in order to solve a problem. Which of the following alternatives would you use? Briefly explain. Shorter, brief, explanations will receive more credit.

```
// Alternative One                          // Alternative Two
public class B extends A {                  public class B {
}                                               A a = new A();
                                            }
```

8. (3 pts) A method called **sequence** throws a checked exception. How would you update the method so we don't have to add a try/catch block?

```
public void sequence(){
    // code that throws checked exception
}
```

9. (3 pts) The signature of the clone method in the Object class is **protected Object clone().** If you are overriding this method in a class called Hero, which of following is the preferred signature for the method in the Hero class?

a. public Object clone()
b. public Hero clone()
c. protected Object clone()
d. protected Hero clone()

10. (7 pts) Define a class called **StringComp** that implements the Comparator interface and compares String objects. The class will allow us to sort strings based on the string's lengths (shorter strings will appear first after sorting an ArrayList of strings).

Problem #2 (Class Implementation)

For this problem you will complete implementation of two classes, **Engine** and **GasEngine**, whose partial definition is provided below. An Engine has a make, mileage, and suppliers that manufacture the engine. There is one supplier designated as the worst supplier for that engine. A **GasEngine** keeps track of the octane required by the engine. All methods of both classes are non-static unless specified otherwise. On the next page you will see a sample driver that illustrates the functionality of the classes you need to implement. Feel free to ignore it if you know what to implement. Also feel free to ignore the ElectricEngine class you see (you do not have to implement it). **Students in the honor section: there is another question for you at the end of the exam.**

```
public abstract class Engine implements Iterable<String>, Comparable<Engine> {
	protected String make;
	private int mileage, numSuppliers;
	private String worstSupplier, suppliers[];
	private static int MAX_SUPPLIERS = 10;
}

public class GasEngine extends Engine {
	private int octane;
}
```

Engine Class Methods

1. Define a constructor that takes as parameters make, mileage, and worstSupplier as parameters initializing the corresponding instance variables. The method will create a String array of length MAX_SUPPLIERS initializing the number of suppliers to 0.
2. Define a method named **getMileage** that returns the mileage.
3. Define a method named **addSupplier** that adds a supplier to the suppliers array. If the argument is **null**, the exception I**llegalArgumentException** will be thrown with the message "invalid supplier" and no further processing will take place. The method returns a reference to the current object. You can assume we will not add more than MAX_SUPPLIERS.
4. Provide any method(s) necessary for the implementation of the **Comparable<Engine>** interface. Engines will be compared based on mileage and those with a higher mileage will appear first if we were to sort a list of engines.
5. Using anonymous inner classes, implement any method(s) necessary for the implementation of the Iterable<String> interface. The iterator will allow us to iterate over the suppliers associated with an engine. The iterator will return the next supplier that is different than worstSupplier (the iterator will skip the worstSupplier). You can assume only one worstSupplier has been added to the list.
6. Define an abstract method named **start()** that has void as return type.

GasEngine Methods

1. Define a constructor that takes as parameters the make, mileage, worstSupplier, and octane as parameters initializing the corresponding instance variables.
2. Define a constructor that takes as parameters the make and mileage as parameters, initializing the corresponding instance variables. The worstSupplier value will be "DSUPPLIER" and octane will be set to 0. You must rely on the previous constructor to implement this one.
3. Define a method named **getOctane** that returns the octane.
4. The **start** method will just have the following code: System.out.println("Starting gas engine: " + make);

Utilities Class Static Method

An **Utilities** class defines the method **getSortedByMileage** that has the signature below. The method will return an ArrayList with GasEngines that have at least an octane value that corresponds to minimum_octane. The ArrayList returned must be sorted by mileage. Feel free to use Collections.sort to sort your data. **Notice that the engines array parameter can have engines other than GasEngines.**

```
public static ArrayList<GasEngine> getSortedByMileage(ArrayList<Engine> engines, int minimum_octane)
```

Driver

```
public static void main(String[] args) {
        ArrayList<Engine> all = new ArrayList<>();
        int mileage = 10, octane = 80, voltage = 220;

        GasEngine ge1 = new GasEngine("CPEng", mileage, "worstA", octane);
        ge1.addSupplier("BestSupplierA").addSupplier("worstA");
        ge1.addSupplier("RegionalSupplier");

        mileage = 15;
        ElectricEngine ee2 = new ElectricEngine("SilSprEng", mileage, "worstSupplierE", voltage);
        ee2.addSupplier("LocalSupplier");

        mileage = 25;
        octane = 90;
        GasEngine ge3 = new GasEngine("BetEng", mileage, "worstA", octane);
        ge3.addSupplier("LocalSupplier");

        all.add(ge1);
        all.add(ee2);
        all.add(ge3);

        for (Engine engine : all) { engine.start(); }

        System.out.print("Suppliers: ");
        for (String supplier: ge1) {
                System.out.print(supplier + " ");
        }
        System.out.println();

        int minimum_octane = 10;
        ArrayList<GasEngine> sorted = getSortedByMileage(all, minimum_octane);
        System.out.println(sorted);
}
```

Output

```
Starting gas engine: CPEng
Starting electric engine: SilSprEng
Starting gas engine: BetEng
Suppliers: BestSupplierA RegionalSupplier
[BetEng, 25, CPEng, 10]
```

PAGE FOR YOUR ANSWERS

PAGE FOR YOUR ANSWERS

PAGE FOR YOUR ANSWERS

HONORS

1. Students in the Honor's section must answer this question.
2. Only students in the Honor's section will receive credit for answering this question.

Implement the clone method for the **Engine** class previously described. The returned object should such that modifications to this object will not affect the original one. You will lose credit if you use deep copy when it is not necessary.