

Part I. Multiple Choice (4 pts)

Choose the single best answer to each of the following statements. You do not need to show your work.

1. In Object-Oriented Programming, which of the following statements correctly describes the relationship between *objects*, *classes*, and *methods*?
 - a. An *object* is a single instance of a *class*. A *class* describes the attributes and *methods* that can be used to interact with any object that belongs to that class.
 - b. An *object* is a general template that includes attributes and *methods*. A *class* is an "instance" of an *object*.
 - c. A *method* is a way of interacting with a *class*. It is defined in the *object* description.
 - d. *Classes* and *methods* are defined in the *object* description.

The Car class describes a vehicle that can drive a given number of miles with a `drive` method, and can have its total miles travelled checked with a `getMiles` method.

2. Which of the following would be a reasonable name for an *instance variable* for the Car class?
 - a. `mySUV`
 - b. `milesTraveled`
 - c. `getMiles`
 - d. `drive`

Continuing with the Car class, consider the following instructions:

```
Car mySUV = new Car();  
mySUV.drive(25);  
System.out.println(mySUV.getMiles());
```

3. Which of the following is a *mutator* method?
 - a. `mySUV`
 - b. `milesTraveled`
 - c. `getMiles`
 - d. `drive`
4. Which of the following is an *accessor* method?
 - a. `mySUV`
 - b. `milesTraveled`
 - c. `getMiles`
 - d. `drive`

Part II. Free Response

1. “Java is an *object-oriented* language, with *classes*. A *class* is an *abstraction*, and usually uses *encapsulation*. The *public interface* for a program is used by other programmers, although *instance variables* are almost always *private*.”

Explain what these terms mean.

2. The `Person` class describes a human being in terms of two instance variable: their name and their age (an `int` value). A constructor takes these two values as parameters to initialize the instance variables.

There are four methods that have been defined for the class:

- i. `getName()` - returns the value of the name
- ii. `getAge()` - returns the person's age
- iii. `changeName()` - takes a new name as a parameter and updates the person's name
- iv. `celebrateBirthday()` - adds one to the person's age

In the space below, write a `PersonTester` file with a `main` method that:

- a. Creates a new `Person` object with a name and age of your choosing
- b. Uses `Person` methods to print out the person's name and age
- c. Calls methods to change the person's name and celebrate their birthday
- d. Print out the person's updated name and age

```
public class PersonTester
{
    public static void main(String[] args)
    {
        // Response for part (a)

        // Response for part (b)

        // Response for part (c)

        // Response for part (d)

    }
}
```

3. The `ElectronicBattery` class describes a battery that can be used in electronic devices, and recharged when its energy level gets too low. Two instance variables are used in this class:
- a. `maxHoursOfCharge` keeps track of the total number of hours of charge that the battery can hold. This value doesn't change, and is defined by a double parameter in the `ElectronicBattery` constructor.
 - b. `remainingCharge` describes the current charge of the battery as it gets used. When the battery is fully-charged, this variable has a value equal to `maxHoursOfCharge`. When the battery is completely used up, it has a value of 0. This value is initially set to 0 when an `ElectronicBattery` instance is constructed.

The class includes three methods:

- a. `getHoursRemaining()` - returns the number of hours of charge remaining in the battery
- b. `recharge()` - restores the battery to its maximum possible charge
- c. `useBattery()` - takes a double parameter `hoursUsed` and modifies the object accordingly. If `hoursUsed` is greater than the remaining charge left in the battery, the battery will be completely used up and a "Recharge needed" message printed out.

Write the complete `ElectronicBattery` class.