Set 11 Study Questions

1. What is a Java interface?

2. What is polymorphism?

3. Suppose you have an interface called `CanDance`, and three classes (`Student`, `Penguin`, and `Cow`) all of which implement the `CanDance` interface. Also assume that there is a method available with the following prototype:

```
public static doSquareDance(CanDance a)
```

Decide which of the following code fragments are reasonable:

```
a. CanDance x = new CanDance();
b. CanDance y = new Student();
c. Student z = new CanDance();
d. Student z = new Penguin();
e. CanDance a;
     a = new Student();
     a = new Penguin();
     a = new Cow();
f. Penguin b = new Penguin();
     doSquareDance(b);
g. Student c = new Student();
     doSquareDance(c);
h. Cow d = new Cow();
     doSquareDance(d);
i. CanDance e = new Student();
     doSquareDance(e);
```

4.

a. Write a `Car` class. (Use your imagination.)

b. Now write an interface called `_CanFixCars_` with two method prototypes:

```
public void fixFlat(Car c);
```

```
public void fixRadiator(Car c);
```

c. Write three classes: `CSMajor, MathMajor`, and `CEMajor`, each of which implements the `_CanFixCars_` interface. Be creative when implementing the methods. How do you think a Math Major would fix a flat tire? ☺

d. In a separate class, write a static method with the following prototype:

```
public static fixCar(Car c, CanFixCars repairPerson)
```

The method should somehow determine what is wrong with the car (is it a flat tire, a broken radiator, or something else) and have the repair person fix the car by calling the repair person_s `fixFlat` or `fixRadiator` methods.

e. Finally, write a main method that will create several broken cars, create several students of various kinds, and have the students fix the cars. (I.e.: make several calls to your `fixCar` method.)

5. What is meant by the term _algorithm_?

6. Name several problems that can be solved with just an algorithm.

7. Name several problems that are too complicated to be solved with just a single algorithm.

8. What is a _use case_? Imagine that you are working on online banking program. Describe several _use cases_ that your program should be able to deal with. (Recall that there are three parts to the description of a _use case_: the pre-conditions, the actions, and the post-conditions.)