

Write a class called LargestDimension that implements the interface below to return the largest dimension of Rectangle objects. (note: this is similar to the assignment with the PerimeterMeasurer class)

```
public interface Measurer
{
    double measurer(Object obj);
}
```

The class you implemented above is sometimes called a callback class - what are some of the advantages of a callback class?

Consider the Java code below & determine what the final output is

```
int x=0;
for(int a=5; a>=0; a--)
{
    for(int b=0; b<a; b++)
    {
        x++;
    }
}
System.out.println("x: " + x);
```

List and briefly define the 6 stages of the software life cycle. Where applicable, indicate the outputs (aka deliverables) of the stages.

Consider the following code & determine the output:

```
public class Person
{
    private String name;

    public Person(String n)
    {
        name = n;
    }
    public void changeName(String n)
    {
        name = n;
    }
    public String getName()
    {
        return name;
    }
}
. . .

Person p1 = new Person("smith");
Person p2 = p1;
p1.changeName("jones");
System.out.println( p2.getName() );
```

Consider a class called HybridCar that inherits from a class called Car & also implements an interface called HybridTechnology. Draw out the UML diagram that illustrates the relationships between HybridCar, Car and HybridTechnology.

Referring to the example above, indicate which classes, if any, are super classes & which classes, if any, are subclasses.

What type of programming uses classes called listeners? Given an example of software that uses this style of programming & explain your answer.

When we print out objects from the Rectangle class & BankAccount class, we get something like the following:

```
"java.awt.Rectangle[x=5,y=10,width=20,height=30]"
```

```
"BankAccount@d24606bf"
```

What is the name of the method that is responsible for printing out the object's information?

Which class is overriding the given method?

Where does the other class inherit the method from?

Given two classes, CheckingAccount & SavingsAccount which both inherit from a class BankAccount, place an --> by the statements that will print:

```
CheckingAccount ca = new CheckingAccount(100.0);
SavingsAccount sa = new SavingsAccount (200.0);
BankAccount ba = new BankAccount (300.0);
Object o1 = (Object)ca;
Object o2 = (BankAccount)o1;

if(ca instanceof CheckingAccount)
    System.out.println("ca instanceof CheckingAccount");

if(sa instanceof CheckingAccount)
    System.out.println("sa instanceof BankAccount");

if(ba instanceof BankAccount)
    System.out.println("ba instanceof BankAccount");

if(ba instanceof Object)
    System.out.println("ba instanceof Object");

if(o1 instanceof Object)
    System.out.println("o1 instanceof Object");

if(o2 instanceof CheckingAccount)
    System.out.println("o2 instanceof CheckingAccount");
```

Consider the following class

```
public class RegularPerson
{
    private String name;
    public RegularPerson (String n)
    {
        name = n;
    }
    public String getName()
    {
        return name;
    }
}
```

Make a new class called KnightedPerson that inherits from regular person. Assume the following bit of code will be used to test your class:

```
KnightedPerson k = new KnightedPerson("Jim", "Moorhead");
System.out.println( k.getName() );
System.out.println( "expected: Sir Jim of Moorhead" );
```

Make certain that you include documentation comments for your class & all of its methods/constructors.

Describe a similarity and a difference between the waterfall & spiral software development models:

What is the difference between the "uses" & aggregation relationships? Which of the two is considered stronger & why?

When developing classes for a problem, nouns usually indicate \_\_\_\_\_, while verbs or actions indicate \_\_\_\_\_.

In Java a given class may implement at most a single interface.  
True            False

A shallow copy of a given object creates complete copies of its subobjects.  
True            False

The operator instanceof works on both classes & interfaces.  
True            False