PGCert IT: Programming for Industry

Inheritance

Exercise One: Scanner and Printer

The following two interfaces specify the functions that a Document Scanner and a Printer are expected to have:

```
public interface Scanner {
	public Document getDocument();
	public boolean jobsDone();
	public Error getError();
}

public interface Printer {
	public void printDocument(Document d);
	public int getEstimateMinutesRemaining();
	public Error getError();
}
```

For the following two classes write down the methods that they must implement.

```
public class PrintOmatic implements Printer {
	// TODO Implement the necessary methods
}

public class OverpricedAllInOnePrinterfier implements Scanner, Printer {
	// TODO Implement the necessary methods
}
```

Which of the following lines are valid?

1. `Scanner eg2 = new PrintOmatic();`
2. `Printer eg3 = new OverpricedAllInOnePrinterfier();`
3. `OverPricedAllInOnePrinterfier eg4 = new Printer();`
4. `Printer eg1 = new Scanner();`

Exercise Two: Simple Animal Class

The following interface specifies the functions that `IAnimal` is expected to perform:

```
public interface IAnimal {
	// Returns a string containing the greeting
	public String sayHello();

	// Returns true or false;
	public boolean isMammal();

	// Returns the name, followed by "the" followed by the
	// animal type, e.g. "George the Monkey"
	public String myName();

	// Returns the number of legs
	public int legCount();
}
```

Here is an example output of the application:

```
Tweety the bird says tweet tweet.
Tweety the bird is a non-mammal.
Did I forget to tell you that I have 2 legs.
-------------------------------------------------------------
Bruno the dog says woof woof.
Bruno the dog is a mammal.
Did I forget to tell you that I have 4 legs.
-------------------------------------------------------------
Mr. Ed the horse says neigh.
Mr. Ed the horse is a mammal.
Did I forget to tell you that I have 4 legs.
This is a famous name of my animal type: PharLap
-------------------------------------------------------------
```

Complete the `Bird` and `Dog` classes that implement the interface `IAnimal`. Complete the `Horse` class that implements the interface `IAnimal` and another interface `IFamous` which will be used to tell the user about the famous kiwi horse `"PharLap"`.

```
public interface IFamous {
	// What is a famous name for this animal
	public String famous();
}
```

Now create a simple application and call the method `processAnimalDetails(IAnimal[] list)` which iterates through an array of animals and gives the example output. Note that this method will call `myName`, `sayHello`, `isMammal`, and `legCount` for each one of the animals. We also like to print famous names of animals if they exist. Hint: use `instanceof` operator.

Exercise Three: Polymorphism

1. What is the output when you run the following code?

```
public class SuperClass {
	public int x = 10;
	static int y = 10;

	SuperClass() {
		x = y++;
	}

	public int foo() {
		return x;
	}

	public static int goo() {
		return y;
	}
}

public class Test1 extends SuperClass {
	int x2= 20;
	static int y2 = 20;

	Test1() {
		x2 = y2++;
	}

	public int foo2() {
		return x2;
	}

	public static int goo2() {
		return y2;
	}

	public static void main(String[] args) {
```

```
		SuperClass s1 = new SuperClass();
		Test1 t1 = new Test1();
		System.out.println("The Base object");
		System.out.println("S1.x = " + s1.x);
		System.out.println("S1.y = " + s1.y);
		System.out.println("S1.foo() = " + s1.foo());
		System.out.println("S1.goo() = " + s1.goo());
		System.out.println("\nThe Derived object");
		System.out.println("\nInherited fields");
		System.out.println("T1.x = " + t1.x);
		System.out.println("T1.y = " + t1.y);
		System.out.println("T1.foo() = " + t1.foo());
		System.out.println("T1.goo() = " + t1.goo());
		System.out.println("\nThe instance/class fields");
		System.out.println("T1.x2 = " + t1.x2);
		System.out.println("T1.y2 = " + t1.y2);
		System.out.println("T1.foo2() = " + t1.foo2());
		System.out.println("T1.goo2() = " + t1.goo2());
	}
}
```

2. What is the output when you run the following code? The `SuperClass` will remain the same.

```
public class Test1 extends SuperClass {
	static int x = 15;
	static int y = 15;
	int x2= 20;
	static int y2 = 20;

	Test1() {
		x2 = y2++;
	}

	public int foo2() {
		return x2;
	}

	public static int goo2() {
		return y2;
	}

	public static int goo(){
		return y2;
	}
```

```
        public static void main(String[] args) {
              SuperClass s2 = new Test1();
              System.out.println("\nThe static Binding");
              System.out.println("S2.x = " + s2.x);
              System.out.println("S2.y = " + s2.y);
              System.out.println("S2.foo() = " + s2.foo());
              System.out.println("S2.goo() = " + s2.goo());
        }
  }
```

3. To which class is the method `s2.goo()` called?

4. What is the static type of variable `s2`?

5. Are we able to make a call to method `foo2()` from variable `s2`?

6. What is the result from the following line of code?

   ```
   Test1 t2 = new SuperClass();
   ```

7. What is the result from the following line of code?

   ```
   Test1 t2 = (Test1) new SuperClass();
   ```