AP Computer Science

Inheritance Basics Homework

(For #2, we haven't talked about overloading in a long time. Here's the difference: http://www.programcreek.com/2009/02/overriding-and-overloading-in-java-with-examples/)

1. What is code reuse? How does inheritance help achieve code reuse?

2. What is the difference between overloading and overriding a method?

3. Consider the following classes:

```
public class Vehicle {...}

public class Car extends Vehicle {...}

public class SUV extends Car {...}
```

Which of the following are legal statements?

a. `Vehicle v = new Car();`
b. `Vehicle v = new SUV();`
c. `Car c = new SUV();`
d. `SUV s = new SUV();`
e. `SUV s = new Car();`
f. `Car c = new Vehicle();`

Interacting With the Super Class Homework

4. When are appropriate times to use the **super** keyword?

5. For the next three problems, consider the following class:

```
public class Student {
        private String name;
        private int age;

        public Student(String n, int a) {
                name = n;
                age = a;
        }

        public void setAge(int a) {
                age = a;
        }
}
```

Also consider the following partial implementation of a subclass of `Student` to represent undergraduate students at a university:

```
public class UndergraduateStudent extends Student {
    private int year;
    ...
}
```

Can the code in the `UndergraduateStudent` class access the `name` and `age` fields it inherits from `Student`? Can it call the `setAge` method?

6. Write a constructor for the `UndergraduateStudent` class that accepts a name as a parameter and initializes the `UnderGraduateStudent`'s state with that name, an `age` value of `18`, and a `year` value of `0`.
7. Write a version of the `setAge` method in the `UndergraduateStudent` class that not only sets the `age` but also increments the `year` field's value by one.

Polymorphism Homework

8. Using the classes A, B, C, and D defined below...

```
1  public class A {
2      public void method1() {
3          System.out.println("A 1");
4      }
5
6      public void method2() {
7          System.out.println("A 2");
8      }
9
10     public String toString() {
11         return "A";
12     }
13 }
```

```
1  public class B extends A {
2      public void method2() {
3          System.out.println("B 2");
4      }
5  }
```

```
1  public class C extends A {
2      public void method1() {
3          System.out.println("C 1");
4      }
5
6      public String toString() {
7          return "C";
8      }
9  }
```

```
1  public class D extends C {
2      public void method2() {
3          System.out.println("D 2");
4      }
5  }
```

What is the output of the code below?

```
public static void main(String[] args) {
    A[] elements = {new B(), new D(), new A(), new C()};
    for (int i = 0; i < elements.length; i++) {
        elements[i].method2();
        System.out.println(elements[i]);
        elements[i].method1();
        System.out.println();
    }
}
```

13. Assume that the following classes have been defined:

```
1  public class Bay extends Lake {
2      public void method1() {
3          System.out.print("Bay 1 ");
4          super.method2();
5      }
6      public void method2() {
7          System.out.print("Bay 2 ");
8      }
9  }
```

```
 1 public class Pond {
 2     public void method1() {
 3         System.out.print("Pond 1 ");
 4     }
 5     public void method2() {
 6         System.out.print("Pond 2 ");
 7     }
 8     public void method3() {
 9         System.out.print("Pond 3 ");
10     }
11 }
```

```
1  public class Ocean extends Bay {
2      public void method2() {
3          System.out.print("Ocean 2 ");
4      }
5  }
```

```
1  public class Lake extends Pond {
2      public void method3() {
3          System.out.print("Lake 3 ");
4          method2();
5      }
6  }
```

What output is produced by the following code fragment?

```
Pond[] ponds = {new Ocean(), new Pond(), new Lake(), new Bay()};
for (Pond p : ponds) {
    p.method1();
    System.out.println();
    p.method2();
    System.out.println();
    p.method3();
    System.out.println("\n");
}
```

14. Suppose that the following variables referring to the classes from the previous problem are declared:

```
Pond var1 = new Bay();
Object var2 = new Ocean();
```

Which of the following statements produce compiler errors? For the statements that do not produce errors, what is the output of each statement?

```
((Lake) var1).method1();
((Bay) var1).method1();
((Pond) var2).method2();
((Lake) var2).method2();
((Ocean) var2).method3();
```

Interfaces Homework

19. What is the difference between implementing an interface and extending a class?

20. Consider the following interface and class:

```
public interface I {
    public void m1();
    public void m2();
}
public class C implements I {
    // code for class C
}
```

What must be true about the code for class `C` in order for that code to compile successfully?

21. What's wrong with the code for the following interface? What should be changed to make a valid interface for objects that have colors?

```
public interface Colored {
    private Color color;
    public Color getColor() {
        return color;
    }
}
```

Abstract Classes Homework

24. What is an abstract class? How is an abstract class like a normal class, and how does it differ? How is it like an interface?

25. Consider writing a program to be used to manage a collection of movies. There are three kinds of movies in the collection: dramas, comedies, and documentaries. The collector would like to keep track of each movie's title, the name of its director, and the year the movie was made. Some operations are to be implemented for all movies, and there will also be special operations for each of the three different kinds of movies. How would you design the class(es) to represent this system of movies?

Design Homework

15. What is the difference between an is-a and a has-a relationship? How do you create a has-a relationship in your code?

16. Imagine a `Rectangle` class with objects that represent two-dimensional rectangles. The `Rectangle` has `width` and `height` fields with appropriate accessors and mutators, as well as `getArea` and `getPerimeter` methods.

You would like to add a `Square` class into your system. Is it a good design to make `Square` a subclass of `Rectangle`? Why or why not?

17. Imagine that you are going to write a program to play card games. Consider a design with a `Card` class and 52 subclasses, one for each of the unique playing cards (for example, `NineOfSpades` and `JackOfClubs`). Is this a good design? If so, why? If not, why not, and what might be a better design?