COMP 249: Object Oriented Programming II

Tutorial 2:
Intro to Inheritance

Overriding Methods

- Consider the following two classes:

```
public class Dog {
    public void bark() { ... }
    public void wagTail() { ... }
    public static void sleep(int minutes) { ... }
}

public class BullDog extends Dog {
    public static void bark() { ... }
    public void wagTail() { ... }
    public void sleep(int minutes) { ... }
}
```

- Which method(s) overrides a method in the superclass? What happens to the other method(s)?

Accessing Parent Methods From the Child Class

- Consider the following two classes:

```
public class Dog {
    public String toString() {
        return "This is a dog";

    }
}
public class Bulldog extends Dog { }
```

What would be the output of the following:

```
Dog fido = new Dog();
Bulldog terror = new Bulldog();
System.out.println(fido);
System.out.println(terror);
```

Overriding Methods

- Consider the following two classes:

```
public class Dog {
    public String bark() {
        return "Bark!";
    }
    public String bark2() {
        return "Bark! Bark!";
    }
}
public class Chiwawa extends Dog {
    public String bark2() {
        return "Yip! Yip!";
    }
}
```

What would be the output of the following:

```
Dog fido = new Dog();
Chiwawa carlos = new Chiwawa();

System.out.println(fido.bark());
System.out.println(carlos.bark());
System.out.println(fido.bark2());
System.out.println(carlos.bark2());
```

Accessing Overridden Methods of the Superclass

- Consider the following two classes:

```
public class Dog {
    public String toString() {
        return "This is a Dog";

    }
}
public class Bulldog extends Dog {
    public String toString() {
        return super.toString() + " but also a Bulldog";

    }
}
```

What would be the output of the following:

```
Dog fido = new Dog();
Bulldog terror = new Bulldog();
System.out.println(fido);
System.out.println(terror);
```

A more complete example

- Consider the following class definition:

```
public class Card {
    private String recipient = "";
    private String occasion = "";
    public String getRecipient() {return recipient;}

    public void setRecipient(String recipient) {
        this.recipient = recipient;
    }
    public String getOccasion() { return occasion;}

    public void setOccasion(String occasion) {
        this.occasion = occasion;
    }
    public Card(String recipient, String occasion){
        this.recipient = recipient;
        this.occasion = occasion;
    }
    public void greeting(){ System.out.println("Happy "+ occasion);}
}
```

We will now extend this class...

A more complete example (Birthday)

```
class BirthDay extends Card {
    private int age;
    public BirthDay(String recipient, int age) {
        super(recipient,"Birthday");
        this.age = age;
    }

    public void greeting() {
        System.out.print("Dear " + getRecipient() + " ");
        super.greeting();
        System.out.println("Happy " + age + "th Birthday\n\n");
    }
}
```

A more complete example (Holiday)

```
class Holiday extends Card {
    public Holiday(String recipient) {
        super(recipient, "Holiday");
    }
    public void greeting() {
        System.out.println("Dear " + getRecipient());
         super.greeting();
    }
}
```

A more complete example (Valentine)

- Finally, we extend the Valentine class:

```
class Valentine extends Card {
    private int kisses;
    public Valentine(String r, int k) {
        super(r,"Valentine");
        kisses = k;
    }
    public void greeting() {
        System.out.println("Dear " + super.getRecipient() +" ");
        super.greeting();
        System.out.println("Love and Kisses,\n");
        for (int j=0; j < kisses; j++)
            System.out.print("X");
        System.out.println("\n\n");
    }
}
```

A more complete example (Driver)

- Using the classes just described, what would be the output of the following:

```
Card crd = new Card("Luncinda","Holiday");
crd.greeting();
Valentine crd2 = new Valentine("Walter", 7);
crd2.greeting();
```

String

- Consider the following code:

```
class StringTest {

    private static void modifyString(String s){
        s = "fedcba";
    }

    public static void main(String[] args){
        String s = "abcdef";
        modifyString(s);
        System.out.println(s);
    }
}
```

What would be the output?

Coding Exercise (1)

- Let's build a package of Shapes based on this first class:

```
package shapes;
public class Shape {
    private int height = 0;
    private int width = 0;
    private char pattern = '*';
    // Constructors
    public Shape() {}
    public Shape(int h, int w) {
        this.height = h;
        this.width = w;
    }
}
```

Coding Exercise (2)

```
// Setters
public void setHeight(int h) {
    this.height = h;
}
public void setWidth(int w) {
    this.width = w;

}
public void setPattern(char c) {
    this.pattern = c;

}
```

Coding Exercise (3)

- // Getters

```
public int getHeight() {return this.height;}
public int getWidth() {return this.width;}
public char getPattern() {return this.pattern;}
public String toString() {
    return getClass() + " => Height: " +
            getHeight() + " Width: " + getWidth();
}
```

Coding Exercise

- Using the Shape.java file available on moodle (or the tutor distributes the code), extend the Shape class into the following classes:
- Rectangle: Define the draw() method to print a rectangle of height by width of the char stored in pattern
- Square: Override both setWidth and setHeight so that both values are always the same. Write the appropriate draw() method.
- Triangle: Add a variable angle which will be set to 90 (for simplicity's sake). Override toString() to print the angle as well as the rest of the information. Implement the draw() method.