# COMP 6481

**Tutorial 1:**

Review and Inheritance

# Question 1

Assume a rectangle class with two attributes, a and b representing the size of the rectangle. What is the output of this code,

```
public void myMethod(Rectangle rect) {
    rect.a = 15;
    rect.b = 15;
}

public static void main(String[] args) {
    Rectangle r = new Rectangle(10, 10);
    MyClass c = new MyClass();

    c.myMethod(r);
    System.out.println(r.toString()); // Rectangle size
}
```

# Question 2

What is the output of this code, assuming previous rectangle class:

```
public void myMethod(Rectangle rect1, Rectangle rect2) {
    rect1 = rect2;
}

public static void main(String[] args) {
    Rectangle r1 = new Rectangle(10, 10);
    Rectangle r2 = new Rectangle(15, 15);
    MyClass c = new MyClass();

    c.myMethod(r1, r2);
    System.out.println(r1.toString()); // Rectangle size
    System.out.println(r2.toString()); // Rectangle size
}
```

# Question 3: Consider these two classes

```java
public class Animal {
    private int age;
    private String name, color;

    public Animal(int age, String name, String color) {
        this.age = age;
        this.color = color;
        this.name = name;
    }

    public String toString() {
        return "Animal: Name: " + this.name + ", Age: " +
                this.age + ", Color: " + this.color;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setColor(String color) {
        this.color = color;
        }
}
```

```java
public class House {
    private String address;
    private Animal animal;

    public House(String address, Animal animal) {
        this.address = address;
        this.animal = animal;
    }

    public String toString() {
        return "House: Address: " + this.address
                + ",Contains: " + this.animal;
    }

    public String getAddress() {
        return this.address;
    }

    public Animal getAnimal() {
        return this.animal;
    }
}
```

# Question 3:

What would be the output of:

```
class driver {
    public static void main(String[] arg) {
        Animal a = new Animal(2, "Emma", "Red");
        House h1 = new House("Montreal", a);
        a.setAge(3);
        a.setName("Liam");
        a.setColor("White");
        House h2 = new House("Toronto", a);
        System.out.println(h1 + "\n" + h2);
    }
}
```

# Comments in Java

3 types of comments in Java:

- `// single line comments`

- ```
  /* Multiple lines comment.
  Useful to "erase" a block
  of code from compilation */
  ```

- ```
  /**
  * JavaDoc comments
  * Can be used to generate html documentation
  */
  ```

# Inheritance

- 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();
```