CS2030 Programming Methodology
Semester 1 2018/2019

31 August 2018
Tutorial 1
Abstraction, Encapsulation and Inheritance

1. Given the following program fragment.

```
class A {
    public int x = 5;
    public static int y = 1;

    public A() {
        x = x + 1;
        y = y + 1;
    }
}
```

By either creating a `main` method or using `JShell`, invoke the following:

```
A a1 = new A();
A a2 = new A();
```

(a) After executing `a1.x = 10`, what is the value of `a2.x`?

(b) After executing `a1.y = 10`, what is the value of `a2.y`?

(c) What is the significance of the `static` keyword used during instance variable declaration? How is it useful?

(d) Is `A.x = 3` a valid statement? How about `a1.x = 3` and `A.y = 3`?

2. Consider the following two classes:

```
public class P {
    private int x;
    public void changeSelf() {
        x = 1;
    }
    public void changeAnother(P p) {
        p.x = 1;
    }
}
```

```
public class Q {
    public void changeAnother(P p) {
        p.x = 1;
    }
}
```

(a) Which line(s) above violate the private access modifier of `x`?

(b) What does this say about the concept of an "abstraction barrier"?

3. Given the following class `Circle`.

```
public class Circle {

    Point centre;
    double radius;

    public Circle(Point centre, double radius) {
        this.centre = centre;
        this.radius = radius;
    }

    @Override
    public boolean equals(Object obj) {
        System.out.println("equals(Object) called");
        if (obj == this) {
            return true;
        }
        if (obj instanceof Circle) {
            Circle circle = (Circle) obj;
            return (circle.centre.equals(centre) && circle.radius == radius);
        } else {
            return false;
        }
    }

    public boolean equals(Circle circle) {
        System.out.println("equals(Circle) called");
        return circle.centre.equals(centre) && circle.radius == radius;
    }
}
```

In the following fragment

```
Circle c1 = new Circle(new Point(0, 0), 10);
Circle c2 = new Circle(new Point(0, 0), 10);
Object o1 = c1;
Object o2 = c2;
```

What is the output of the following statements?

(a) `o1.equals(o2);`

(b) `o1.equals((Circle) o2);`

(c) `o1.equals(c2);`

(d) `o1.equals(c1);`

(e) `c1.equals(o2);`

(f) `c1.equals((Circle) o2);`

(g) `c1.equals(c2);`

(h) `c1.equals(o1);`

4. Which of the following program fragments will result in a compilation error?

   (a)
   ```
   class A {
       public void f(int x) {}
       public void f(boolean y) {}
   }
   ```

   (b)
   ```
   class A {
       public void f(int x) {}
       public void f(int y) {}
   }
   ```

   (c)
   ```
   class A {
       private void f(int x) {}
       public void f(int y) {}
   }
   ```

   (d)
   ```
   class A {
       public int f(int x) {
           return x;
       }
       public void f(int y) {}
   }
   ```

   (e)
   ```
   class A {
       public void f(int x, String s) {}
       public void f(String s, int y) {}
   }
   ```

5. In Lecture #3, we designed the class `Rectangle` that inherits from the class `Shape`. Now we want to design a class `Square` that inherits from `Rectangle`. A square has the constraint that the four sides are of the same length.

   (a) How should `Square` be implemented such that we obtain the following using `JShell`?

   ```
   jshell Shape.java Rectangle.java Square.
   java
   |  Welcome to JShell -- Version 9.0.4
   |  For an introduction type: /help intro

   jshell> Square s = new Square(5)
   s ==> Square with area 25.00 and perimeter 20.00

   jshell>
   ```

   (b) Do you think `Square` should inherit from `Rectangle`? Or should it be the other way around? Or maybe they should not inherit from each other?