ICS4U Module 4: Note & Exercise 1a

What is an Object?

- An *object*:
    - Stores data
    - Can perform actions
    - Provides communication
    - Has a *state*: the data it stores
- Has *behaviours*: defined by the action and communication it provides - Objects model or simulate real-world things: we will use an example – a Circle object

Object: Circle

- A circle object is defined by its radius, because in real life, the radius defines a circle - Some circle actions:
    - Change its radius
    - Calculate its radius
    - Tell us its radius
- An object is an **instance** of a class
- The **class** is a data type that defines
    - Variables for the state of an object
    - Methods for an object's behaviour

Encapsulation:

- We only change the state of an object through its behaviour
    - This means we only use methods to change an object
    - For ex – we change the radius using a method, probably, changeRadius() - Also called *information hiding*

Client Code

- Client code refers to an application that uses one or more classes
- The client can access the methods, but does not access the data

directly Here is some client code that uses the Circle class:

```
public class TestCircle {

      public static void main(String[]
      args) {
            Circle spot = new Circle();

            spot.setRadius(5);

            System.out.println("Circle radius:" +
            spot.getRadius());
```

a Circle object named spot is instantiated

Output:

Circle radius: 5.0
Circle area: 78.5

```
            System.out.println("Circle area: " + spot.area());
        }
}
```

ICS4U Module 4: Note & Exercise 1a

Designing & Writing a Class

- When designing a class, decide:
    - The data the object will store
    - The actions and communication the object will provide
    - Variable names, and method names
    - A description of the method, with parameters

Circle class design:

Circle Class Design:
variables: radius, PI
methods:
setRadius – changes the radius. Requires one parameter, for radius
getRadius – returns the circle radius
area – returns the area of the circle based on the current radius

- A class is written in a separate file (client code and classes are compiled together in a single project)
- Includes
    - A declaration: includes access level, keyword class, and the class name
    - A body: contains:
        - Variables
        - Constructors (used to initialize variables)
        - Methods

Variables and methods are called the members of a class

- The general form of a class:

```
<access_level> class <name> {
   <variables>
   <constructors>
   <methods>
}
```

Three types of class methods:

1) Accessor Method: called to determine the value of a variable
2) Modifier Method: called to change the value of a variable
3) Helper Method: called from within a class by other methods, should have access level `private`

ICS4U Module 4: Note & Exercise 1a

class name should be a noun, begin with an uppercase letter, no spaces

Implementation of the Circle Class

```
/*
 * Circle class
 */

public class Circle {
	private static final double PI = 3.14;
	private double radius;

	/**
	 * constructor
	 * pre: none
	 * post: A Circle object created. Radius initialized to 1.
	 */
	public Circle() {
		radius = 1; //default radius
	}

	/**
	 * Calculates the radius of the circle
	 * pre: none
	 * post: Radius has been changed
	 */
	public void setRadius(double newRadius) {
		radius = newRadius;
	}

	/**
	 * Calculates the area of the circle.
	 * pre: none
	 * post: The area of the circle has been returned
	 */
	public double area() {
		double circleArea;

		circleArea = PI * radius * radius;
		return(circleArea);
	}

	/**
	 * Returns the radius of the circle
	 * pre: none
	 * post: The radius of the circle has been returned
	 */
	public double getRadius() {
		return(radius);
	}

}
```

member variables are declared before, and outside of any methods

this is a modifier method

this is an accessor method

ICS4U Module 4: Note & Exercise 1a

Programming Exercise:

a) Copy and paste the Circle class provided here into your editor. Modify the Circle class to include a member method named `circumference`. The circumference() method should return the circumference of the circle (2****). Test the class with the following client code.

```
public class TryCircle {


   public static void main(String[] args) {
         Circle spot = new Circle();

         spot.setRadius(3);
         System.out.println("Circle radius: " + spot.getRadius());
         System.out.println("Circle circumference: " +
    spot.circumference()); }
}
```

b) Create a Coin class that includes a variable `faceUp` that stores either a 0 for heads up or 1 for tails up, an accessor method named showFace() that returns a 0 if the coin is heads up or a 1 if the coin is tails up, and a modifier method named flipCoin() that assigns a random integer between 0 and 1 inclusive, to the variable `faceUp`. Test the class with the following client code:

```
/*
 * TestCoin.java
 * Coin--part 1 of 2
 *
 */

 /**
 * The Coin class is tested.
 */
 public class TestCoin {

      public static void main(String[] args) {
            Coin nickel = new Coin();

            nickel.flipCoin();
            if (nickel.showFace() == 0) {
                  System.out.println("Heads up!");
            } else {
                  System.out.println("Tails up!");
            }
       }

 }
```

Do not submit your code for either part a or part b just yet.

```
public class TestCoin {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Coin nickel = new Coin();
		nickel.flipCoin();
		if (nickel.showFace() == 0) {
		System.out.println("Heads up!");
		} else {
		System.out.println("Tails up!");
		}



	}

}
class Coin {
	int flipcoin;
	public int  flipCoin() {
		flipcoin = (int)Math.round(Math.random());
		return flipcoin;
	}

	public int showFace() {
```

```
                if (flipcoin == 0) {
                        return 0;
                }
                else {
                        return 1;
                }
        }
}
```