Objects and Classes

Encapsulation

COMP2603
Object Oriented Programming 1

Week 2, Lecture 2

Outline

- Message Passing Syntax
  - Method signatures
  - Method overloading
- Types of Variables
  - Class, instance variables
  - Variable scope
- Types of Methods
  - Class, instance methods
- ArrayLists

Method Signature

A method causes certain actions to take place and can be regarded as a service provided by an object.

The method signature consists of the method name, and the input parameter types.

```
public double getUnit( ){
    return unit;
}
public int getUnit(String s){
    return unit;
}
```

Method Overloading

The process of writing methods in the same class with the same name but with different method signatures.

The 2 methods in the following example are said to be overloaded.

Example 1

```
public class TemperatureSensor{
     //Attributes
     private double temperature;
     private final String unit = “Celcius”;

     //Accessors
     public double getTemperature(){
          return temperature
     }
     public double getTemperature(String units){
        // return temperature in F or C as specified
     }
     public String getTemperature(String units, double precision){
        // return temperature in F or C specified as a String
        // to a particular precision
     }
}
```

Overloaded Constructors

A class may have more than one constructor with different types and numbers of input parameters.

These constructors are said to be overloaded since they all share the same name (class name) and allow an instance of the class to be created in different ways.

Example 2

```
public class Book {
     //Attributes
     private String title;
     private double price;
     //Assume accessors & mutators

     //Constructors
     public Book(){
        title = null;
        price = 0;
     }
     public Book(String title){
        this.title = title;
        price = 0;
     }
     public Book(String title, double price){
        this.title = title;
        this.price = price;
     }
}
```

```
public class BookDemo {


      public static void main(String[] args){
       Book b1 = new Book();
       Book b2 = new Book("Flowers for Algernon");
       Book b3 = new Book("The BFG", 15.00);



        }
     }
```

Instance Variables

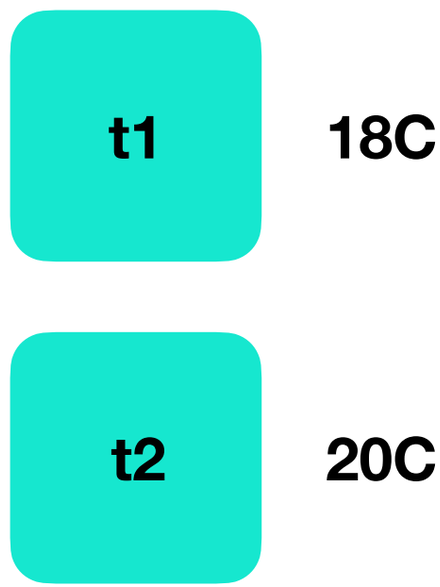
A repository for part of the state of an object.

The terms instance variable and object variable are interchangeable.

Collectively, the instance variables of an object constitute its structure.

example:

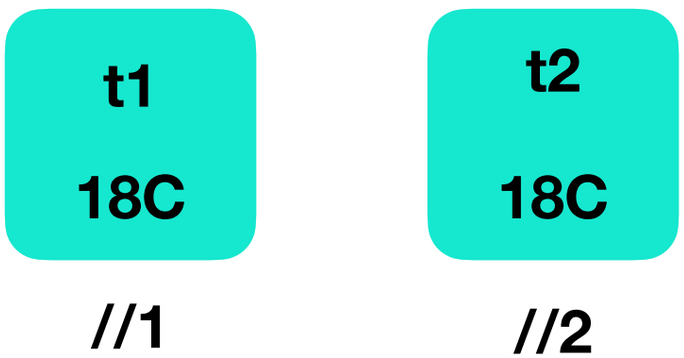
```
private double temperature;
```

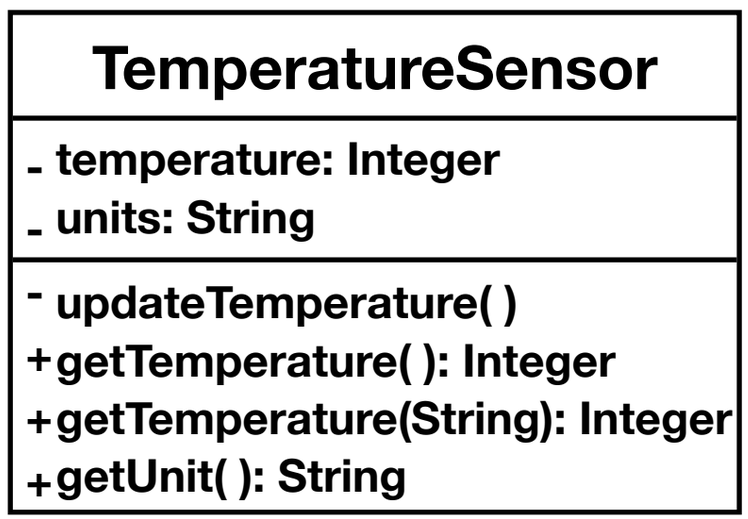


```
public class SensorApplication{
  public static void main(String[] args){

      TemperatureSensor t1 = new TemperatureSensor(); //1
      TemperatureSensor t2 = t1;                      //2
      t2.updateTemperature();                         //3

    }
}
```





Class Variables

Part of the state of a class. Collectively, the class variables of a class constitute its structure.

A class variable is shared by all instances of the same class.

example:

```
private static int sensorIDCounter;
```

Example

```
public class TemperatureSensor{
    private static int sensorIDCounter = 1;

    private double temperature; //instance variable
    private int sensorID;

    public TemperatureSensor(){
       temperature = Math.random();
       sensorID = sensorIDCounter;
       sensorIDCounter++;
    }

}
```

```
public class SensorApplication{
  public static void main(String[] args){

      TemperatureSensor t1 = new TemperatureSensor(); //1
      TemperatureSensor t2 = new TemperatureSensor(); //2

    }
}
```

sensorID= 1

t1

18C

//1

sensorIDCounter = 2

sensorID= 2

t2

18C

//2

sensorIDCounter = 3

TemperatureSensor
- temperature: Integer **- units: String**
- updateTemperature( ) **+getTemperature( ): Integer** **+getTemperature(String): Integer** **+getUnit( ): String**

Instance Methods

- Instance Methods: These require an instance of a class to exist before they can be invoked.

example:

```
public double getTemperature(){..}

TemperatureSensor t1 = new TemperatureSensor();
double temp = t1.getTemperature();
```

Class Methods

- Class Methods: These provide a service even when no instance of the class is available

example:

```
public static void main(String[] args){..}
```

```
public static void setIDCounter(){
    sensorIDCounter = 100;
  }
```

Class Methods

- Class Methods:
  - cannot refer to instance instance variables
  - can refer to variables declared locally within the method

example:

```
public static void setIDCounter(){
    sensorIDCounter = 100;
  }

  TemperatureSensor.setIDCounter(); //usage
```

References

- Mohan, Permanand (2013) FUNDAMENTALS OF OBJECT-ORIENTED PROGRAMMING IN JAVA