Object Oriented Programming – SCJ2153

Java Variables

Associate Prof. Dr. Norazah Yusof

Variables

- Variables are used to store data in a program.
- There are two types of variables in Java:
  1. Primitive variables
  2. Reference variables

Primitives Variables

- A primitive can be one of eight types: char, boolean, byte, short, int, long, double, or float.
- Once a primitive has been declared, its primitive type can never change, although in most cases its value can change.
- Primitive variables can be declared as **class variables (statics)**, **instance variables**, **method parameters**, or **local variables**.
- One or more primitives, of the same primitive type, can be declared in a single line.
- Examples of primitive variable declarations:
  - `char oneLetter;`
  - `boolean flag;`
  - `int x, y, z;`
  - `double area;`

Reference Variables

- A reference variable is used to refer to an object.
- A reference variable is declared to be of a specific type and that type can never be changed.
- Reference variables can be declared as **static variables**, **instance variables**, **method parameters**, or **local variables**.
- One or more reference variables, of the same type, can be declared in a single line.
- Examples of reference variable declarations:
  - `Object obj;`
  - `Employee newEmployee;`
  - `String myAddress;`

Instance Variables

- Instance variables are defined inside the class, but outside of any method, and are only initialized when the class is instantiated.
- Instance variables are the fields that belong to each unique object.
- For example, the following code defines fields (instance variables) for the radius of circle objects:

```
public class Circle extends Object  //Class header
{
    private double radius;
```

- Each circle instance will know its own radius. In other words, each instance can have its own unique values for this field.
- The term "field," "instance variable," "property," or "attribute," mean virtually the same thing.

Local Variables

- Local variables are variables declared within a method.
- Its life starts inside the method and destroyed when the method has completed.
- Local variables are always on the stack, not the heap.
- Before the content of a local variable can be used, it **must be assigned** with a value.
- The following print statement may cause error:

```
class TestLocalVar {
     public void myMethod() {
       int counter;
       System.out.println (counter);
     }
    }
```

Local Variables (cont.)

- A local variable can't be referenced in any code **outside the method** in which it's declared.
- The statement `counter = i` below, will not compile because the variable `counter` is refered outside of method `myMethod()`.

```
class TestLocalVar {
  public void myMethod() {
    int counter = 10;
  }
  public void yourMethod(int i) {
   counter = i;
  }
}
```

- However, the value of `counter` can be passed out of the method to take on a new life.

Constant

- **Constant** represent permanent data that will never change.
- To declare a constant need to use the `final` keyword.
- Syntax to declare a constant:
- ```
  final datatype CONSTANTNAME = value;
  ```
- For example:

```
final double PI = 3.14159; //PI as constant
 final int SIZE = 3;
```
```

Constant(cont.)

- Once a constant has already assigned a value, it cannot be assigned a new value .
- The statement `MAX_SIZE = 100` below is illegal because it tries to give new value to the constant .

```
class TestConstant {
  final int MAX_SIZE = 50;

  public void myMethod() {
     MAX_SIZE = 100;
  }
}
```