Core Java 8

Lesson 01: Declaration And Access Control

Capgemini

Lesson Objectives

After completing this lesson, participants will be able to -

- Declare Class in Java
- Develop Small Programs in Java

Identifiers And JavaBeans

Identifiers are the names of variables, methods, classes, packages and interfaces. Unlike literals they are not the things themselves, just ways of referring to them.

JavaBeans:

JavaBeans are Java classes that have properties.
For our purposes, think of properties as private instance variables.
Since they're private, the only way they can be accessed from outside of their class is through methods in the class.
The methods that change a property's value are called setter methods,
The methods that retrieve a property's value are called getter methods

Legal Identifiers :

Rules for Legal Identifiers:

- Identifiers must start with a letter, a currency character ($), or a connecting character such as the underscore (_). Identifiers cannot start with a number!
- After the first character, identifiers can contain any combination of letters, currency characters, connecting characters, or numbers.
- In practice, there is no limit to the number of characters an identifier can contain.
- You can’t use a Java keyword as an identifier.
- Identifiers in Java are case-sensitive; foo and FOO are two different identifiers

Examples of Legal Identifiers :

MyVariable	_myvariable
MYVARIABLE	sum_of_array
Myvariable	geeks123
X	$myvariable
i	
x1	

Sun's Java Code Conventions

Why Have Code Conventions :

Code conventions are important to programmers for a number of reasons:

- 80% of the lifetime cost of a piece of software goes to maintenance.
- Hardly any software is maintained for its whole life by the original author.
- Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.
- If you ship your source code as a product, you need to make sure it is as well packaged and clean as any other product you create.

Java uses Camel Case as a practice for writing names of methods, variables, classes, packages and constants.

Camel case in Java Programming : It consists of compound words or phrases such that each word or abbreviation begins with a capital letter or first word with a lowercase letter, rest all with capital.

Sun’s Java Code Conventions

Classes and Interfaces :

Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Interfaces name should also be capitalized just like class names.

Use whole words and must avoid acronyms and abbreviations.

Examples :

interface Bicycle ,Class MountainBike ,Class Football, Class Circle

Methods :

Methods should be **verbs**, in mixed case with the **first letter lowercase** and with the first letter of each internal word capitalized.

Examples :

void changeGear(int newValue);

void speedUp(int increment);

void applyBrakes(int decrement);

JavaBeans Standards

A bean itself must adhere to the following conventions:

Class name

There are no restrictions on the class name of a bean.

Superclass

A bean can extend any other class. Beans are often AWT or Swing components, but there are no restrictions.

Instantiation

A bean must provide a no-parameter constructor or a file that contains a serialized instance the beanbox can deserialize for use as a prototype bean, so a beanbox can instantiate the bean. The file that contains the bean should have the same name as the bean, with an extension of .ser.

Declare Classes

Class:

- A template for multiple objects with similar features
- A blueprint or the definition of objects

Object:

- Instance of a class
- Concrete representation of class

```
class < class_name>
{
    type var1; …
    Type method_name(arguments )
     {
         body
       } …
} //class ends
```

Source File Declaration Rules

Each Java source file contains a single public class or interface. When private classes and interfaces are associated with a public class, you can put them in the same source file as the public class.

The public class should be the first class or interface in the file.

Java source files have the following ordering:

- Beginning comments
- Package and Import statements; for example:

```
import java.applet.Applet;

import java.awt.*;

import java.net.*;
```

- Class and interface declarations

Class Declaration And Access Modifiers

Modifiers

- default
- private
- public
- protected
- static
- final

```
public class Person {

    private String firstName;
    private String lastName;
    protected String address;
    public int age ;
    String mobileNo;

}
```

Concrete Subclass

A concrete class in java is any such class which has implementation of all of its inherited members either from interface or abstract class. Lets take an example:

```
public abstract class A {
     public abstract void methodA();
}
interface B {
     public void printB();
}
public class C extends A implements B {
       public void methodA() {
            System.out.print("I am abstract implementation");
        }
      public void printB() {
           System.out.print("I am interface implementation");
       }
}
```

Declaring an Interface

Like a class, an interface can have methods and variables, but the methods declared in interface are by default abstract (only method signature, no body).

Interfaces specify what a class must do and not how. It is the blueprint of the class.

An Interface is about capabilities like a Player may be an interface and any class implementing Player must be able to (or must implement) move(). So it specifies a set of methods that the class has to implement.

If a class implements an interface and does not provide method bodies for all functions specified in the interface, then class must be declared abstract.

```
interface <interface_name> {
 //declare interface fields that are constants
//declare methods that are abstract
}
```

Declaring Interface Constants

interface data members are limited to **static final variables**, which means that they are constant. An object variable can be declared as an interface type, and all the constants and methods declared in the interface can be accessed from this variable. All objects, whose class types implement the interface, can then be assigned to this variable.

```
public interface Testable {
void method1();
void method2(int i, String s);
int x=10;
}
```

Declare Class Members

A Java class can contain the following building blocks:

- Fields
- Constructors
- Methods
- Nested Classes

Class with Fields :

```
public class Car {

    public String brand ;
    public String model ;
    public String color  ;

}
```

Class with Constructors:

```
public class Car{

 private String brand;
 private String model ;
 private String color;

 public Car(){
}
 public Car(String brand, String model, String color){
 ……
 …..
}
```

Declare Class Members

Class with Methods :

```
public class Car{
public String brand;
public String color;
public String model;
 public void setColor(color c){
  color=c;
}
}
```

Access Modifiers

- Default
- Private
- Public
- Protected

Location/Access Modifier	Private	Default	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Non-Access Modifiers

Java provides a number of non-access modifiers to achieve many other functionality.

- The *static* modifier for creating class methods and variables.
- The *final* modifier for finalizing the implementations of classes, methods, and variables.
- The *abstract* modifier for creating abstract classes and methods.
- The *synchronized* and *volatile* modifiers, which are used for threads.

Variable Declarations

You must declare all variables before they can be used. Following is the basic form of a variable declaration −

Data type variable [= value][,variable [=value] ..] ;

Here *data type* is one of Java's datatypes and *variable* is the name of the variable. To declare more than one variable of the specified type, you can use a comma-separated list.

Example

```
int a, b, c;         // Declares three ints, a, b, and c.
int a = 10, b = 10;  // Example of initialization
byte B = 22;         // initializes a byte type variable B.
double pi = 3.14159; // declares and assigns a value of PI.
char a = 'a';        // the char variable a iis initialized with value 'a'
```

Variable Declarations

There are three kinds of variables in Java –

- Local variables
  - Local variables are declared in methods, constructors, or blocks
  - Local variables are created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor, or block.

- Instance variables
  - Instance variables are declared in a class, but outside a method, constructor or any block.
  - When a space is allocated for an object in the heap, a slot for each instance variable value is created.

- Class/Static variables
  - Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
  - There would only be one copy of each class variable per class, regardless of how many objects are created from it.

Summary

In this lesson, you have learnt:

- Sun’s Java coding conventions
- Class Declaration
- Interface declaration
- Access Modifiers
- Enums
- Writing, Compiling, and Executing a simple program

Review Question

Question 1:Correct output for code is :
 enum color { red, green, blue }
color c; c = color.red;
 System.out.println(color);

1. 1
1. -1
1. red
1. 0