Java – Objects & Classes

Java supports the following fundamental concepts:

- Polymorphism
- Inheritance
- Encapsulation
- Abstraction
- Classes
- Objects
- Instance
- Method
- Message Parsing

Object - Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.

Class - A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.

Sample of a class:

```
public class Dog{

	String breed;
	int age;
	String color;

	void barking(){

	}

	void hungry(){

	}

	void sleeping(){

	}
}
```

A class can contain any of the following variable types.

- **Local variables:** Variables defined ***inside methods, constructors or blocks*** are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.

- **Instance variables:** Instance variables are variables ***within a class but outside any method***. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.

- **Class variables:** Class variables are variables declared within a class, outside any method, ***with the static keyword***.

A class can have any number of methods to access the value of various kinds of methods. In the above example, barking(), hungry() and sleeping() are methods.

Constructors

When discussing about classes, one of the most important sub topic would be constructors. Every class has a constructor. If we do not explicitly write a constructor for a class, the Java compiler builds a default constructor for that class.

Each time a new object is created, at least one constructor will be invoked. The main rule of constructors is that they should have the same name as the class. A class can have more than one constructor.

Following is an example of a constructor:

```
public class Puppy{

	public Puppy(){

	}

	public Puppy(String name){
		// This constructor has one parameter, name.
	}

}
```

Creating an Object

As mentioned previously, a class provides the blueprints for objects. So basically, an object is created from a class. In Java, the new keyword is used to create new objects.

There are three steps when creating an object from a class:

- **Declaration:** A variable declaration with a variable name with an object type.
- **Instantiation:** The 'new' keyword is used to create the object.
- **Initialization:** The 'new' keyword is followed by a call to a constructor. This call initializes the new object.

Following is an example of creating an object:

```
public class Puppy{
       public Puppy(String name){
              // This constructor has one parameter, name.
              System.out.println("Passed Name is :" + name );
       }

       public static void main(String []args){
              // Following statement would create an object myPuppy
              Puppy myPuppy = new Puppy( "tommy" );
       }
}
```

Accessing Instance Variables and Methods

Instance variables and methods are accessed via created objects. To access an instance variable, following is the fully qualified path:

```
/* First create an object */
ObjectReference = new Constructor();

/* Now call a variable as follows */
ObjectReference.variableName;

/* Now you can call a class method as follows */
ObjectReference.MethodName();
```

This example explains how to access instance variables and methods of a class

```
public class Puppy{

	int puppyAge;

	public Puppy(String name){
		// This constructor has one parameter, name.
		System.out.println("Name chosen is :" + name );
	}

	public void setAge( int age ){
		puppyAge = age;
	}

	public int getAge(){
		System.out.println("Puppy's age is :" + puppyAge );
		return puppyAge;
	}

	public static void main(String []args){

		/* Object creation */
		Puppy myPuppy = new Puppy( "tommy" );

		/* Call class method to set puppy's age */
		myPuppy.setAge( 2 );

		/* Call another class method to get puppy's age */
		myPuppy.getAge( );

		/* You can access instance variable as follows as well */
		System.out.println("Variable Value :" + myPuppy.puppyAge );
	}
}
```