JAVA

Java is an Object-Oriented programming language most of it is free and open source.

It is developed in the early 1990s, by James Gosling of Sun Microsystems.

**Features of Java:**

* Simple , Robust, Security, Platform independent
* Interpreted and Compiled
* Multithreaded, Dynamic, Ambiguity
* Memory Management and Garbage Collection

**JRE:**

JRE is the “Java Runtime Environment”. It is responsible for creating a Java Virtual Machine to execute Java class files (that is, run Java programs).

**JDK:**

JDK is the “Java Development Kit”. It contains tools for Development of Java code (for example: Java Compiler) and execution of Java code (for example: JRE)

**Data types:**

|  |  |
| --- | --- |
| Primitive | Non primitive |
| Byte, short, int, long, float, double, char, boolean | Class, array, string |

**Operators:**

* Arithmetic (+, -, \*, /, %, ++, --, +=, -=, \*=, /=, %=).
* Bitwise (~,&,|,<<,>>,>>>,&=,|=,^).
* Relational (>=,<=,<,>,==,!=),
* Logical (&&,||,!,^,==,?:).
* *instanceof* Operator.

**Variables:**

Variable is basic storage in a Java program

Three types of variables:

* Instance variables (Instantiated for every object of the class).
* Static variables (Class Variables, Not instantiated for every object of the class).
* Local variables (Declared in methods and blocks).

Example:

public class Box {

private double dblWidth; //Instance Variable

private static int boxid; //Static Variable

public double calcVolume() {

double dblTemp; //Local Variable

}

}

**Control Statements:**

* Use control flow statements to: Conditionally execute statements, Repeatedly execute a block of statements, Change the normal, sequential flow of control
* Categorized into two types:
  + Selection Statements
  + Iteration Statements

**Selection Statements:**

**Switch:**

switch(expression){

case value1: //statement sequence

break;

……………………

default: //default statement sequence

}

**IF:**

if (condition) statement1;

else statement2;

**Iteration Statements:**

**While:**

While(condition)

{

//body

}

**Do-while:**

Do

{//body of loop

}

While (condition);

**For:**

for( initialization ; condition ; iteration)

{ //body of the loop }

**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.

A class may consist the following elements:

* Fields
* Methods
* Constructors
* Initializers

**Default Constructors:**

* All Java classes have *constructors*. Constructors initialize a new object of that type
* Default no-argument constructor is provided if program has no constructors
* Constructors: Same name as the class, No return type, not even void.

**OOPs Concepts:**

**Inheritance:**

Acquiring members from super class to sub class.

Inheritance is ideal for those classes which has “is-a” relationship (using extends keyword).

“Object” class is the ultimate superclass in Java.

**Polymorphism:**

A single entity which perform multiple behaviors.

Java implements polymorphism in two ways

* Method Overloading: A class having multiple methods with same name but different parameters.
* Method Overriding: Multiple methods with same method name with same parameters in two different classes having IS-A relationship.

**Abstract class:**

A class that is declared with abstract keyword, is known as abstract class in java. It can have abstract and non-abstract methods. It cannot be instantiated.

**Interface:**

A Java interface definition looks like a class definition that has only abstract methods, although the abstract keyword need not appear in the definition.

**Exception Handling:**

An exception is an event that occurs during the execution of a program that disrupt its normal course.

General form of exception handling block:

try {//code to be monitored.

}

catch (Exception1 e1 ) {

//exception handler for Type Exception1

}

catch (Exception2 e2 ) {

//exception handler for Type Exception2

}

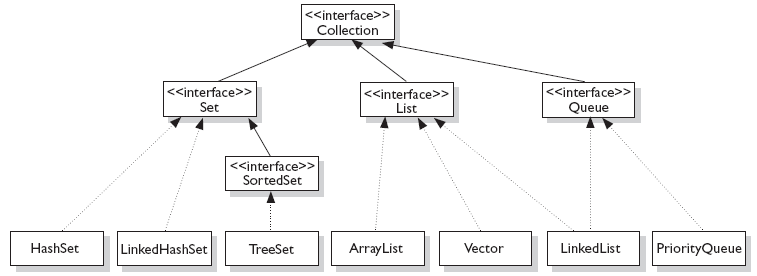
finally {// code that must be executed.

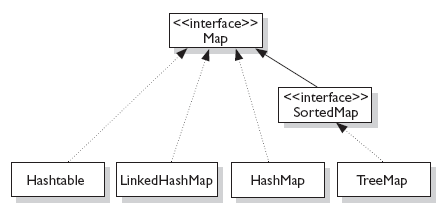
}

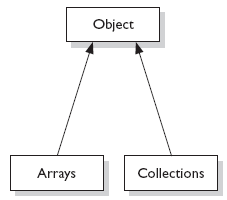
**Keywords for handling exceptions:**

* try : This marks the start of a block associated with a set of exception handlers.
* catch : The control moves here if an exceptions is generated.
* finally :This is called irrespective of whether an exception has occurred or not.
* throws : This describes the exceptions which can be raised by a method.
* throw : This raises an exception to the first available handler in the call stack, unwinding the stack along the way.

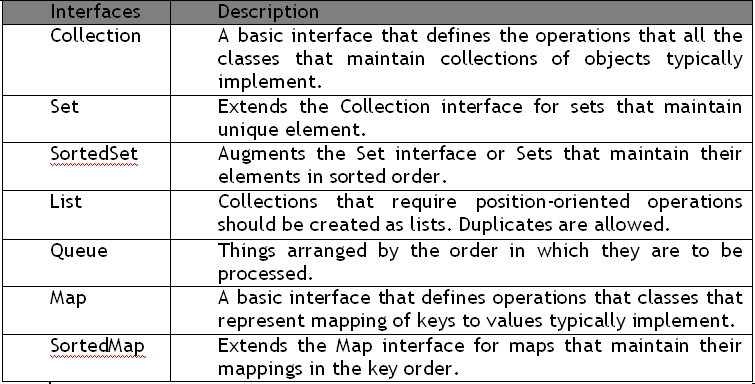
**Collections:**

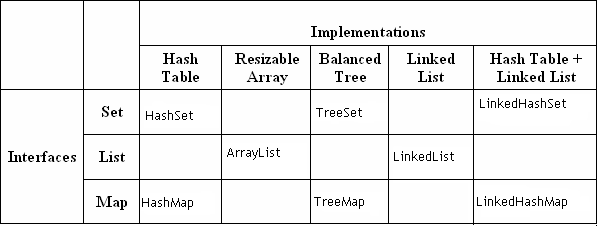


****

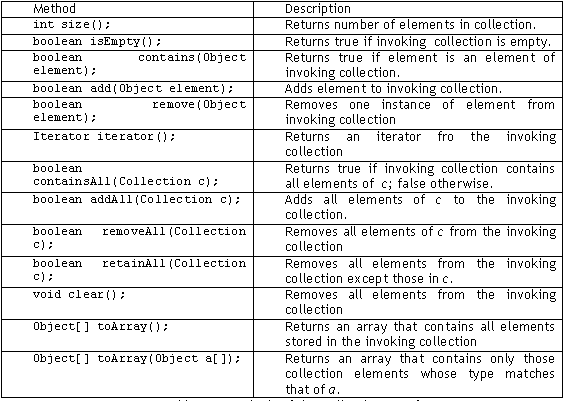


**Collection Interfaces:**



**Collection Implementation:**

**Collection Interface Methods:**



**Property Files:**

Property files have .properties extension and are used to store the configuration parameters. Each parameter is stored as key/value pair

The java.util. Properties class represents a persistent set of key/value properties, are subclasses of Hashtables provides methods to store and retrieve values from properties files.

**Layered Architecture:**

Layering partitions the functionality of an application into separate layers that are stacked vertically. As shown in the figure above, layered architecture enables developer to make changes on one layer without having any side effects on others.

**Presentation Layer:**

The presentation layer contains the components that implement and display the user interface and manage user interaction. This layer includes controls for user input and display.

**Business logic/Service layer:**

After the presentation layer collect the required data from the user and pass it to the business layer, the application can use this data to perform a business process. Use a business layer to centralize common business logic functions and promote reuse.

**Data Access Layer:**

The data access layer should hide the details of data source access. It should be responsible for managing connections, generating queries, and mapping application domain objects to data source structures.

**Data Transfer Objects:**

DTO’s are simply POJO classes that represent application real world entities. These objects are meant to share data between layers of application. These object contains data values and expose them through properties.