Written by

Indracit S

Core Java Concepts & Definitions.

A big big thank you!

This document is just made for reference , revising and to understand concepts with definition.

Java

Java was developed by Sun Microsystems in 1991, now acquired by Oracle. It was developed by James Gosling and Patrick Naughton.

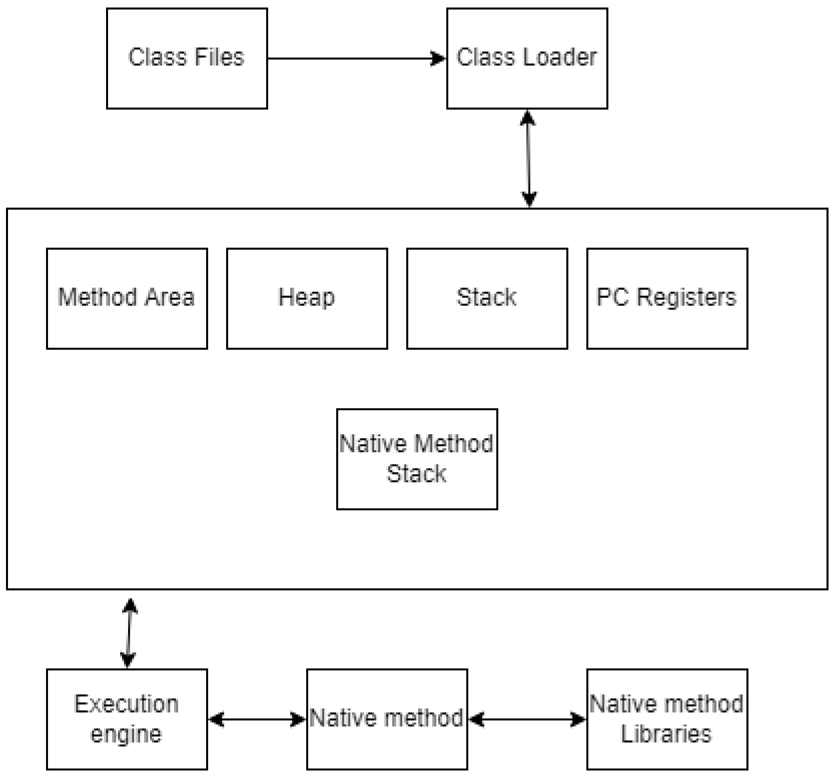
Features of Java

- Platform independent.
- Object oriented Language
- Simple,Robust language, Secure.
- Distributed
- Multi threaded
- Portable

Java virtual Machine

Java is a high level language, program written in high level language cannot run directly on any machine.

Source code(.java file) —----> compiler (javac) ------> Byte code (.class file)



JVM Architecture

How does a JVM work?

Classloader: Class loader reads the .class file and saves the bytecode in the method area.

Method Area: Method area contains class information of all the .class files.

Heap: Heap contains Objects, Heap is part of JVM Memory.

Stack: Part of JVM memory used for storing temporary variables.

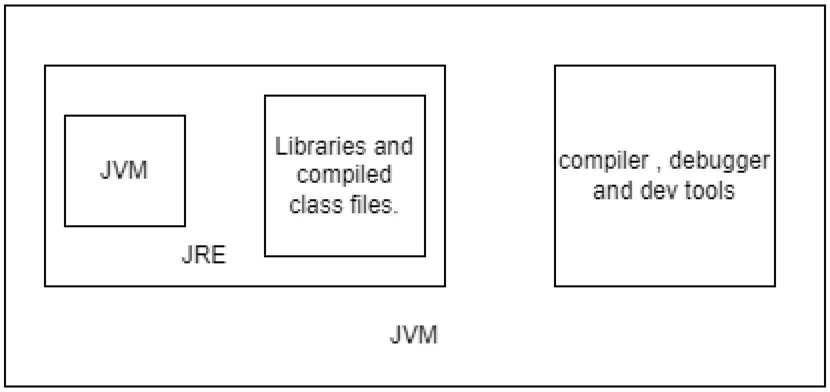
PC registers

It has the track of information of instructions that is executed, Instructions are executed by threads, each thread has separate pc registers.

Garbage collection

After using the created object , the used object is automatically destroyed by garbage collection of the memory management.

JVM, JRE and JDK


JVM
Libraries and compiled class files.
JRE
compiler , debugger and dev tools
JVM

JRE: Runtime environment for JVM, JRE contains JVM, Class libraries.

JVM: JVM runs the program by using classes, libraries and files provided by JRE.

JDK: Superset of JRE, contains JRE and compiler , Debugger.

Variables

Variables is the container for storing data values while executing the java program.

Variable naming convention

- Variable names should not have white space.
- Variable names begin with _ and $.
- Variable names should begin with lowercase.
- Variable names are case sensitive.

Types:

- Static variable
- Local variable
- Instance variable

Static variables

Also known as class level variables. It is associated with class and same for all the instances of class.

Example:

If i create three Objects for the same class and update the value of the static variables with one object. The value of static variables changes for all the three objects.

Instance variables: Each object(instance) of class has its own copy of instance variables.

Example:

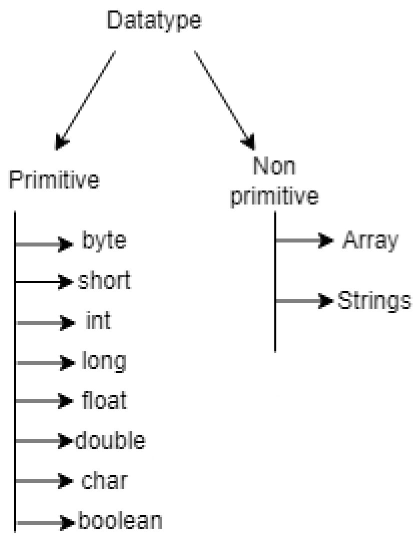
If we create three objects for class and update the instance variables with one object, then for one object the value of the instance variable is changed , and the other two objects have old values.

Local variables

Variables declared inside the method of class , scope is limited to method. You can't change the value or access them outside the method.

Data Types

It defines the type of variable, or defines values that a variable can take. Java is a statically typed language. Static type means datatype of the variable is known at compile time.



Primitive data types

- Size of primitive data types do not differ from one operating system to another.
- Byte,short, int , long used for whole numbers.
- Float and double for fractional numbers.
- Char for characters.
- Booleans show only true or false.

Byte : range -128 to 128 , used to save memory , default size 1 byte, default value 0.

Short : range -32768 to 32767 , size is greater than byte and less than integer, default size 2 bytes, default value 0.

Int : range -2147483648 to 2147483657 , default size 4 bytes , default value 0.

Long : range -93223372036854 to 9223372036854775807 , size 8 bytes , default value 0.

Double : sufficient for holding 15 decimal digits , size 8 bytes.

Float : sufficient for holding 6 to 7 decimal digits, size 4 bytes.

Char : hold alphabets , size 2 bytes**.**

Literals : is a fixed value assigned to the program.

Control flow statement

If we need to execute a set of statements based on the conditions we use control flow statements.

If Statement

If statements consist of a condition followed by a set of statements or statements. Statement will execute if the condition is true.

Nested if: If statement inside if statement is called nested if.

If - else

If block will execute when the condition is true, if the condition is false else block will execute.

If else if : It is used to check multiple conditions.

Switch case

Switch case statement is used when we have a number of choices. We can perform different tasks based on the choice. If the input does not match any of the choices , then the default case will execute. Break is used to bring the control out of the loop.

Points

- Cases don't always need to have order.
- You can use characters.
- The expression inside the switch should result in a constant.
- Nesting switch statements are allowed.

Loops

Loops are used to execute a set of statements until some conditions are met or satisfied.

- For loop
- While loop
- Do while loop

For loop

First : Initialisation.

Second : condition in the for loop is evaluated if it is true , the loop body will be executed.

Third : after every execution of body increment or decrement of loop body executes.

Fourth : condition is reevaluated.

Enhanced for loop is used to iterate over arrays and collections.

While loop

Conditions will be evaluated first and if returns true then the statement inside the loop will be executed.

Note:

If we are using a while loop, then we need to use increment or decrement inside the loop.

Do while loop

Do while is similar to while loop, in while loop condition is evaluated first and loop body is executed next. But in a while loop, the loop body is executed first and condition is evaluated next.

Continue statement

Continue is used inside loops, when it is found in loop control directly jumps to the beginning of the for loop for next iteration , and skip the loop body of the current iteration.

Break statement

Used in 2 scenarios,

Used to come out of the loop instantly.

Used in switch case control, whenever the case is executed if break statement is there means , then it wont execute the subsequent cases.

Object Oriented Programming System

Oops is based on the concept of objects , objects have data , i.e methods and variables. The main purpose of oops is its flexibility and maintainability. Oops brings data and its method in a single location called objects.

What is an object?

Object is a real world entity. It is an instance of class. It is a bundle of variables and its methods. Object has two characteristics: states and behavior. States means variables and behavior means methods.

Ex: House - object

states like address, color , area

Behaviors like open door, close door.

Main principles of oops

- Encapsulation
- Inheritance
- Abstraction
- Polymorphism

What is class in oops?

Class is a collection of objects or blueprint for creating objects.

Ex:

There is a class called website, which has two variables: webname, webpage. We can create any number of objects for this class. We can initialize the two variables webname and webpage for each object. Here the website class is blueprint , by using this class we can create any number of objects.

Constructor

constructor is a block of code that initializes the newly created objects. constructor is not a method, it doesn't have a return type . constructor should have the same name as the class name.

New keyword

New keyword creates object and invokes constructor to initialize newly created object.

this keyword

this keyword refers to the current object.

Types of constructor

- Default constructor
- Noarg constructor
- Parameterized constructor

Default constructor

If you don't implement any constructor , the java compiler inserts the default constructor into your code. This is called the default constructor.

No arg constructor

constructor written by a programmer without arguments is known as no arg constructor.

Parameterized constructor

constructor with arguments is known as a parameterized constructor.

constructor chaining

Calling a constructor from another constructor of the same class is known as constructor chaining. We can write multiple constructor with different parameters and chain them together. By doing this we can do initialisation only once for all these constructor.

super() - refers to the object of immediate parent class.

Whenever the child class constructor is invoked it implicitly invokes the parent class constructor. Compiler inserts a super() statement at the beginning of the child class constructor.

Constructor overloading

constructor overloading is the process of having more than one constructor with different parameters.

static keyword

Static keywords can be with class, variables , method and block. Static members belong to the class instead of a specific instance , this means if you make a member static , you can access it without an object. Static members are common for all the instances of class(object).

Static block

Static block is used for initializing the static variables. Static blocks will get executed when the class is loaded in the memory. A class can have multiple static blocks, which will be executed in the sequence they are written.

Static variables

Static variables are common for all the instances because they are class level variables. Single copy of a static variable is created and shared among all the instances of classes. Memory allocation of static variables is happening once. When class is loaded in the memory. These variables can be accessed directly in static methods.

Static class

A class can be made static only if it is a nested class. Nested static class does not need outer class reference. A static class cannot access non-static members of the outer class.

Inheritance in java

The process by which one class acquires the variables and methods of another class is called inheritance. The class which extends the features of another class is known as child class, subclass or derived class. The class whose properties and functionalities are used by another class is known as parent class, super class or base class.

Advantage of inheritance

- Code reusability
- Code reliability , i.e base class code will be already tested and debugged.
- Less development and maintenance cost.
- Reduce code redundancy

Disadvantages

- Data members in base class are left unused, leading to memory wastage.
- Change in base class will affect.

Extends

To inherit a class we use extends keyword.

Types of inheritance

Single inheritance

Subclass is derived from only one superclass

Multilevel inheritance

A class is derived from another subclass.

Hierarchical inheritance

More than one class is derived from the parent class is called hierarchical inheritance

Multiple inheritance

One class extends more than one class , i.e a child class has two parent classes. Multiple inheritance is not supported in java.

Why is multiple inheritance not supported in java?

Java doesn't allow multiple inheritance to avoid ambiguity. One example of such a problem is the diamond problem.

Class B extends class A, class C extends A , D extends B and C means , if

B and C class have the same method , when we want to use the same method in B and C, which method to be called , method in B or method in C , this leads to ambiguity. That is the main reason java doesn't support multiple inheritance.

Hybrid inheritance

Combination of more than one type of inheritance.

Constructor and inheritance

constructor of the subclass is invoked , when we create an object for the subclass. constructor of subclass invokes the constructor of superclass implicitly or explicitly by the super keyword.

Method Overriding

When we declare a method in child class , which is already present in parent class , it is called method method overriding. We can call the parent class method using the super keyword.

In this case the method in the child class is the overridden method and method in parent class is the overriding method.

Advantage

We can give our own implementation to the inherited method, without even modifying the parent class code.

Rules of method overriding

- Argument list should be same
- Access modifier of the overriding method cannot be more restrictive than the overridden method. Example if the parent class method is public , then the subclass overriding method cannot be private , default or protected.
- Private , static and final methods cannot be overridden.
- If the class extends abstract class or interfaces then all the abstract methods are overridden in the extending class.

IS- A Relationship

Whenever a class inherits another class , it is called IS - A relationship.

HAS - A Relationship

Whenever an instance of one class is used in another class it is called HAS - A relationship.

Method overloading

Class having more than one method with the same name and different parameters.

Types of method overloading

- Changing the number of parameters.
- Changing the datatype of parameters.

- Changing the sequence of data types.

Type promotion

When a datatype of smaller size is converted to datatype of bigger size then it is called type promotion.

Difference between overloading and overriding.

overloading	overriding
Overloading happens at compile time.	Overriding happens at runtime.
Static methods can be overloaded	Static methods cannot be overridden
Overloading happens in same class	Overriding happens in between parent and child class
Private and final methods can be overloaded	Private and final methods cannot be overridden.
Argument list is different	Argument list is same

Polymorphism

Polymorphism means having many forms. We can perform a single action in different ways.

Example

There is a parent class called animal that contains sound() and two subclass horse , cat which extends the parent class animal. These two subclasses override the sound method in animal class. This is an example , where two subclasses perform the same action sound() but in different ways.

Types of polymorphism

- Compile time polymorphism (static polymorphism)
- Runtime polymorphism (dynamic polymorphism)

Compile time polymorphism

Polymorphism that is resolved by the compiler at compile time is called static polymorphism. Method overloading is an example of compile time polymorphism.

Runtime polymorphism

It is known as dynamic method dispatch. Call to overridden method is resolved at runtime. While creating an object parent class is assigned to a subclass object , to determine which method to be called is determined by the type of object.

Abstract class

A class that is declared using abstract keywords is known as abstract class. Abstract classes can have abstract methods as well as concrete methods. A normal class cannot have abstract methods. We cannot create the object for abstract class.

Why do we need abstract classes?

Let's say we have animal classes with a sound method and subclass like lion, tiger , horse etc. each animal has a different sound , so we don't implement the sound method in parent class, so we declare this method as abstract method and class as abstract class.

Rules

- A class derived from abstract class should implement all the abstract methods in abstract class.
- You cannot create the object of abstract class so you should extend a class from abstract class and implement those methods and create objects.
- If the child class does not implement all the abstract methods of the parent class, then the child class also must be declared as abstract.

Abstract method

A method without a body is known as an abstract method.

Interfaces

Interfaces look like class but it is not class. An interface can have methods and variables. But methods are default abstract and variables are public , static and final. Abstract class is used for partial abstraction. Interface is used for full abstraction. Interface cannot implement another interface , it has to extend another interface. Empty interface is known as tag or marker interface. Multiple inheritance is achieved using interfaces.

Packages

Packages is a group of classes , interfaces and other packages.

Types of packages

- Built in packages
- User defined packages

Advantages

- Code reusability
- Better organization
- Prevent from name conflict

Example for built in packages

- java.io.*
- java.lang.*
- java.util.*

Encapsulation

Encapsulation in java is a mechanism of wrapping the variables and methods together in a single unit. The variables are hidden from the users and accessed through the method of the current class. To achieve encapsulation declare variables in private , provide getter and setter methods to view and modify the variables.

Advantages

It improves maintainability, flexibility and reusability. We can make the variable read only or write only by providing any one of the getter and setter methods.

Access modifier

Access modifiers restrict the access of a class , constructor data member and method in another class.

Modifiers in java

- Default
- Private
- Protected
- public

Access modifiers	Class	Package	Subclass (same pack)	Subclass (different pack)	Outside class
public	✔	✔	✔	✔	✔
protected	✔	✔	✔	✔	X
default	✔	✔	✔	X	X
private	✔	X	X	X	X

Garbage collection

The process of removing unused objects from heap areas is known as garbage collection. Garbage collection is done explicitly by calling System.gc().

Exception handling

What is exception handling?

An exception is an unwanted event that interrupts the normal flow of the program , when an execution gets terminated, in such cases we get a system generated error message. These generated messages are not user friendly. By handling exceptions , we can give users meaningful messages.

Advantages

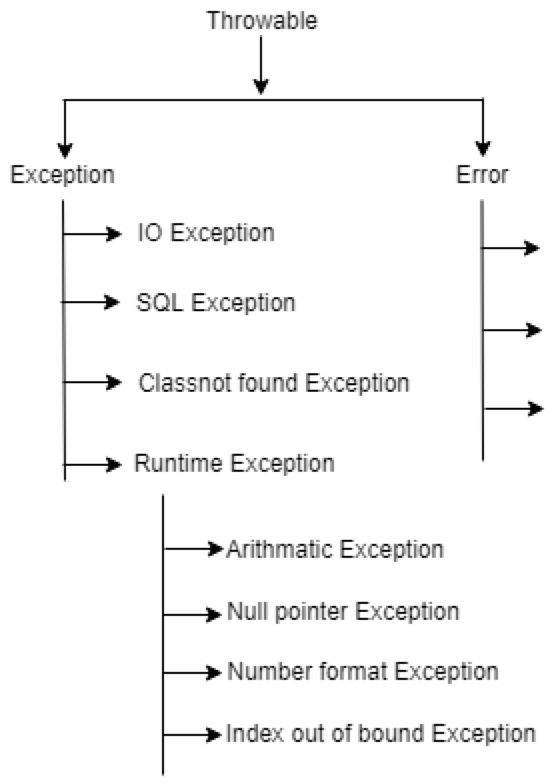
Exception handling ensures the flow of the program doesn't break.

Types of exceptions

- Checked exceptions
- Unchecked exceptions

Hierarchy of java exception classes

The Java.lang.throwable class is the root class of the Java exceptions hierarchy.



Checked exceptions

All exceptions other than runtime exceptions are checked exceptions. Compiler will check if the programmer handled or not , if the programmer does not handle the exception means compilation error will come.

Example

- Sql exception
- IO exception
- Classnot found exception

Unchecked exceptions

Runtime exceptions are also known as unchecked exceptions. These exceptions are not checked at compile time.

Example

- Arithmetic exception
- Null pointer exception
- Array index out of bound exception

Try block and catch block

Try block has a set of statements , where an exception occurs. A try block is always followed by a catch block, and the catch block will handle the exception. Single try block can have multiple catch blocks.

Finally block

Finally the block will always execute, whether the catch block executed or not or try block has exception or not.

Finally block must be associated with a try block,you cannot use finally without try block. Finally is optional. If there is no exception, it means finally will be executed after try block , if any exception occurs means, finally execute after catch.

Finally executes if the try block contains control transfer statements like return , break & continue.

Finally won't execute , when ,

- Death of thread,
- Using system.exit()
- Exception in finally

Throw

Throw keyword is used to throw exceptions explicitly in the method body.

Throws

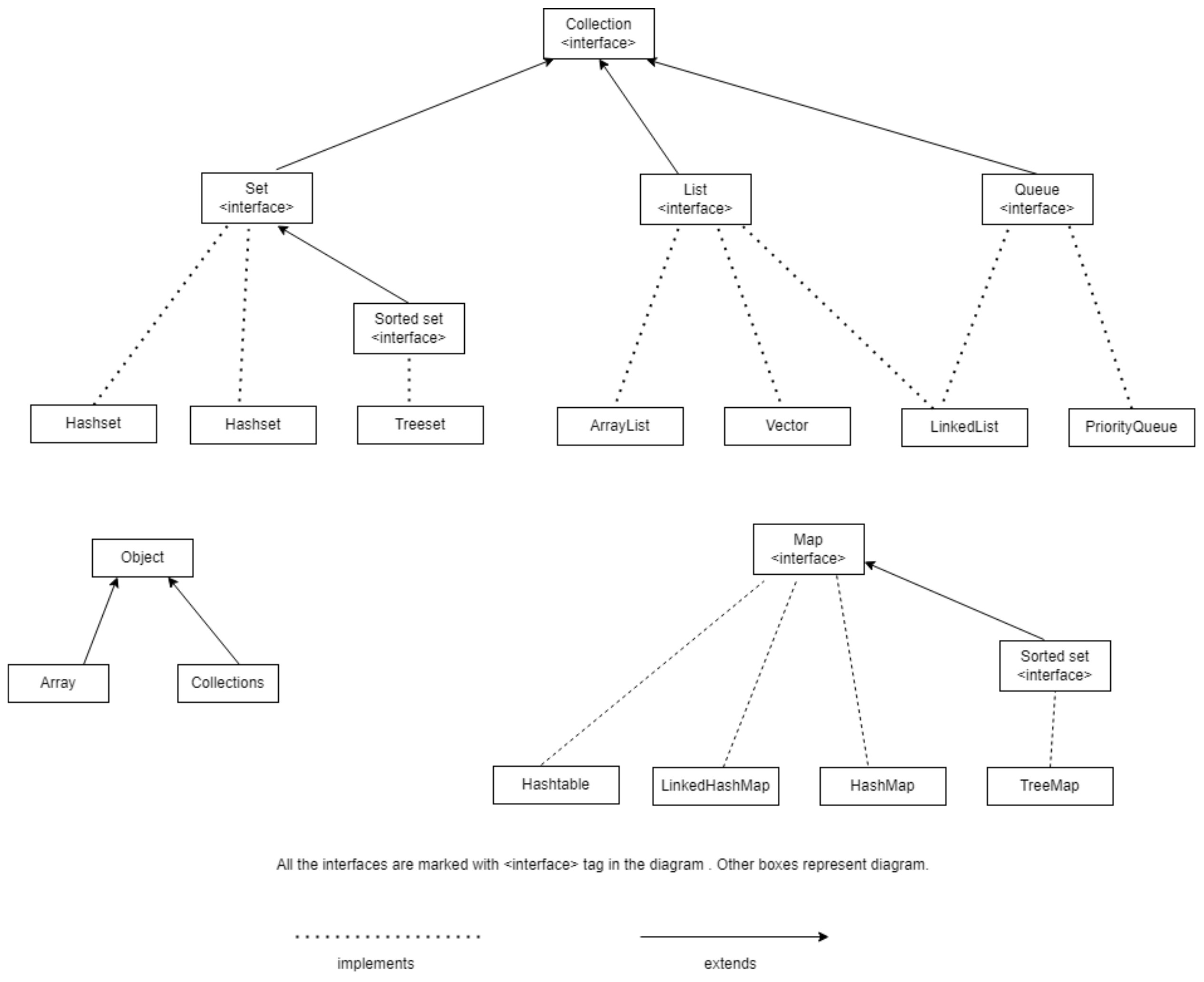
Throws clause is used to declare exceptions in the method signature. You can throw one exception at a time, but you can handle multiple exceptions using throws keyword.

Custom exceptions

We can create our own exception class and throw that exception using throw keyword. These exceptions are called user-defined exceptions.

Collections Framework

Java collections framework is a collection of interfaces and classes which helps in storing and processing data efficiently available in java.util.package. Collection is a container that contains a group of objects. It can grow and shrink during runtime.



All the interfaces are marked with <interface> tag in the diagram . Other boxes represent diagram.

Collection interface

Topmost of collection framework hierarchy. It provides methods for all collection operations.

Methods in collection

add(object obj)	- insert a new object to the collection
clear()	- removes all the element in the collection
removesAll(collection c)	- removes all the objects from current collection.
retainAll(Collection c)	- it deletes the object in the current collection.
isEmpty()	- returns true of the current collection empty.
iterator()	- return the object that is used to iterate over the collection.
size()	- returns no of elements in the current collection.

List

List is ordered collection. List contains duplicate elements. Elements can be inserted and accessed by their position, zero based index.

ArrayList

- It was a dynamic array for storing elements.
- It is similar to an array with no size limits
- It maintains insertion order. It can contain duplicate values; random access is done easily because it works based on index.
- Addition and removal is slower because a lot of shifting is required.
- Initial capacity is 10 and capacity increases 50% when more elements are inserted.

Syntax

```
ArrayList <type> objname = new ArrayList <type> (capacity);
```

LinkedList

- Data is stored as values in doubly linked list format.
- It maintains insertion order.
- It can contain duplicate values.
- Addition and removal can be done fast because no shifting is required.

Syntax

```
LinkedList <type> objname = new LinkedList <type>();
```

Methods

- addFirst()
- addLast()
- getFirst()
- getLast()

Vector

- It is the same as Arraylist.
- It is from java version 1.2 so called legacy class.
- It is mostly used in multithreading , because it is synchronized.
- Capacity increases 100% when more elements are inserted.

Syntax

Vector <type> objname = new Vector <type>();

Stack

Stack is a subclass of vector based on last in first out.

Syntax

Stack <type> objname = new Stack <type> ();

Methods

Boolean empty()	- check if empty or not
Push (element)	- insert element into top of the stack
pop()	- remove and return element from top of the stack
peek()	- return element from top of the stack
search(item)	- return positions of objects in the stack.

Set interface

It does not allow duplicates and indexes

Hashset

Does Not maintain any order. Follows hashing algorithm . Efficient when compared to TreeSet and LinkedHashSet

Syntax

```
HashSet <type> objname = new HashSet <type> ();
```

Methods

- add()
- clear()
- contains()
- isEmpty()
- iterator()
- remove()
- size()

LinkedHashset

It is an implementation class of set interface containing unique elements only. It allows null. Maintains insertion order(what order we give that order will display).

Syntax

```
LinkedHashSet <type> objname = new LinkedHashSet <type>();
```

TreeSet

It implements a set interface and contains unique elements. It does not allow null. It maintains ascending order. Retrieval time is faster when compared to LinkedHashSet.

Syntax

```
TreeSet <type> objname = new TreeSet <> ();
```

Iterator

Iterators are used in collections in java to iterate over elements in collections. The Iterator object is created by calling the iterator method.

Methods

Public boolean hasNext()	- returns true if the iteration has more elements.
Public object next()	- returns the next elements in the iteration
Public void remove()	- remove the next element in the iteration.

Enumeration

Enumerations are used in the collection framework in java to retrieve elements one by one in old classes like vectors.

Methods

Public boolean hasMoreElement()	- returns true if the enumeration has more elements.
Public object nextElements()	- returns the nextElement in the enumeration.

Priority queue

It doesn't follow first in first out. It provides a queue facility. High priority elements will always be in the head of the queue. Display based on the alphabetic order.

Methods

add()	- add element in queue
remove()	- removes and returns element at the head
peek()	- returns element at the head
poll()	- remove the top element and return that element.

Map interface

Map is used to represent key - value pair storage. Each key is mapped at most to one. Each key and value pair is called as an entry. Values are manipulated using the keys.

HashMap

Entries will be arranged based on hashing techniques. Hashmap is unsynchronized. It contains a unique key. It allows null keys and multiple null values. It maintains no insertion order.

LinkedHashMap

It maintains insertion order. It contains values based on the key. It can have one null key and multiple null values. It is unsynchronized. Its initial capacity is 16 and load factor .75

TreeMap

TreeMap is a red - black tree based implementation. It contains values based on key. It implements a navigable map interface and extends abstract map class. Tree map contains unique elements. It is unsynchronized and maintains ascending order.

Comparable interface

Comparable interface is mainly used to sort the array or list of custom objects. List and arrays of objects that implement a comparable interface can be sorted automatically by collections.sort() and arrays.sort().

Comparator interface

By using comparable we can sort any data member, but if we want to sort by multiple choices we can use a comparator interface.

String

String is the sequence of characters. Example hello is a string of 5 characters. String is an immutable object; it is a constant.

Two ways to create string

- String literal
- New keyword

String literal

Assigning string literal to string instance.

String str1 = “welcome”

String str2 = “welcome”

Here we are not using new keywords. Compilers use new keyword and create objects and give to the string instances. If the object already exists means it won't create a new object . It assigns the old object to all the instances.

Example

Here we have two strings welcome . but it creates only one object and assigns to both the instances. If we want two different objects, we have to use a new keyword to create an object.

Multithreading

Thread

Thread is a light - weight smallest part of a process, that concurrently runs with another thread of the same process. Threads are independent because they all have separate paths of execution. That is the reason if an exception occurs in one thread it does not affect the execution of another thread.

All threads share common memories. Process of executing multiple threads simultaneously is known as multithreading. The main purpose of multithreading is to utilize the maximum cpu time.

Life cycle of threads

A thread in java always exists in any one of the following statuses.

- New
- Runnable
- Blocked
- Waiting
- Terminated

New thread

When the thread is created it is in the new state. This thread has not yet started to run.

Runnable thread

A thread is ready to run is moved to this runnable state. In this state, a thread might actually be running or it might be ready to run at any time.

Blocked / waiting state

When the thread is temporarily inactive , then it's in one of the following states.

Blocked : A thread that is blocked waiting for a monitor lock is in this state.

Waiting : A thread that is waiting indefinitely for another thread to perform a particular action is in this state.

Time waiting

A thread that is waiting for another thread to perform an action up to a specific waiting time.

Terminated

A thread that has exited in this state.

Creating a thread in java

- Extending Thread class
- Implementing Runnable Interface

Methods of Thread class

getName()	- It is used for obtaining a thread name.
getPriority()	- obtains a threads priority
isAlive()	- Determine if a thread is still waiting
run()	- entry point for thread
sleep()	- suspend a thread for a period of time
start()	- start a thread by calling its run() method.
join()	- permit one thread to wait until another thread completes its execution.

Thread priorities

Thread priorities are integers which decide how one thread should be treated with respect to the others. Thread priority decides when to switch from one thread to another , a process is called context switching. To set the priority of thread setPriority() method is used which is a method of thread class.Instead of integer for setting priority we can use MIN-PRIORITY , NORM-PRIORITY, MAX-PRIORITY.

Methods : isAlive() and join()

To know whether the thread is finished or not we can use isAlive(), if the thread is not finished means it will give true.

Synchronization

If one thread is writing some data and another thread reading the same data at that time there might be some inconsistency. When two or more threads need to share the same data. There is a way that only resources will be used by one thread at a time. The process to achieve this is called synchronization.

To implement synchronous block, there is synchronous method , thread inside synchronous method is not used by any other thread until the thread comes out of synchronous method.

Inter thread communications

To understand synchronization, java has a concept of monitor. Monitor is a box which held only one thread. If one thread enters the monitor, other threads have to wait. Until the first thread came out of the monitor. Here wait() tells the thread to give up the monitor. Another hand notify() wakes the thread and makes the thread enters monitor. notify() wakes up the first thread that called wait(). notifyall() wakes all the thread that called wait(). Highest priority threads will runs first.

Thread class

Thread class provides constructor and methods to create and perform operations on a thread. Thread class extends object class and extends runnable interface.

Runnable interface

Runnable interfaces are implemented by class, so that the instances of the classes are executed by thread. run() method is used to perform action of thread.

Java Serialization

Serialization is a mechanism to convert an object into a stream of bytes so that it can be written into a file , transported through the network or stored in a database. Java serialization api provides the features to perform serialization and deserialization. Java.io.serializable.

Example

A class implements a serializable interface , all the fields are converted to stream of bytes and accepts the variables which are declared as transient. While doing deserialization , if the stream of bytes contains fields , then the values of fields after deserialization becomes default.

Autoboxing and Unboxing

Autoboxing

Automatic conversion of primitive data types to their corresponding wrapper class.

valueOf() method is used behind the scenes.

Example

Int to integer

Long to long

Char to character.

Unboxing

Opposite to autoboxing , converting objects to their corresponding data type.

intValue() is used in behind the scenes.

Autoboxing happens:

Case 1: when method expecting wrapper class object , but you passed primitive type.

Case 2: when you assign primitive type to the wrapper class object.

Case 3: when dealing with collection.

Wrapper class

Wrapper classes are used to convert primitive data types into objects, like int to integers. In collections , we use generics for type safety, for generics objects are needed, so we are using wrapper class.

Generics in java

Generics means parameterized types. This allows integer, string and user defined types to be the parameter to methods, classes and in interfaces. Using generics , it is possible to create classes with different data types.

Why generics?

Object class is a super class for any other class. So object reference can refer to any object. These lack type safety, generics add type safety.

Types of java generics

Generic method

It take parameters and perform some action and return value. Generic method has type parameters which use actual data type. This allows generic methods to be used in more ways. Compiler takes care of type safety, programmers don't have to perform typecasting.

Generic classes

Generic classes are implemented exactly like non generic classes but it has a type parameter section. There can be more than one parameter separated by comma; this is called parameterized classes.

Type parameters in java:

T - Type

E - Element

K - Key

N - Number

V - value

Advantages

- Code reuse
- Type safety
- Make errors appear at compile time rather than runtime.

File handling in Java

File handling in java permits us to create , read,update and delete the files, which are stored on the local file system. We can perform file handling in java by java I/O api. The Java io package contains all the classes required for input and output operations.

Streams (flow from one place to another)

A stream is a sequence of data in java, a stream is composed of bytes.

Types

- Byte stream
    1. Output stream
    2. Input stream
- Character stream
    1. File writer
    2. File reader

File input stream (Read)

It is used to read data from files as bytes. It is suitable for images, audio and video files.

Syntax

FileInputStream obj = new FileInputStream(filename) ;

Methods

Int read() - return data as byte.

Void close() - to close the file.

Exception - IO exception.

Fileoutput stream

It is used to write data into files as bytes. It is suitable for images, audio and video files.

Syntax

FileOutputStream obj = new FileOutputStream(String name);

Methods

Void write(byte)

Void write(byte[])

Void write(char)

Void close()

Exception - IO exception

Filewriter class

Predefined class in IO package. Used to write data into a file as a character stream. Suitable for text files (.jpg,.txt,.html)

Syntax

FileWriter obj = new FileWriter(String name);

Methods

write(String)

write(byte)

write(char)

close()

FileReader class

Predefined class in IO package. Used to read from a file as a character stream. Suitable for text files.

Syntax

File Reader obj = new File Reader (String name)

Methods

Int read()

close()

File class()

To display a list of files in given folder.

Print writer class

Used to write formatted data.

Syntax

Print Writer obj = new Print Writer

Methods

print(args) - int, double, string

println(arg type)

write(byte)

close()

Console class

Used to read passwords.