CORE JAVA

Method:

It is used to reduce complexity of the code.

Java is platform Independent.

Compiler is used for checking some of the rules and syntactic error.

If there are any errors, compiler will throw runtime error.

If there are no errors, it will convert to .class file.

Java Virtual Machine (JVM)

JRE enables us to create JVM.

Inside JVM, JIT will be present.

If we change the program, then we must recompile it in order to execute it.

.class files contains bytecodes.

Bytecode has made machine platform Independent.

Objects :

Objects are elements of programs that has some data which is known as state.

Objects also have behaviors (methods) which means they can perform certain operations.

Java is object Oriented programming language.

Object has certain states and behavior.

States means data related to some objects.

Behaviour means some methods.

Keywords:

Words which have some predefined meaning.

Java is case Sensitive language.

There are 50 Keywords in java.

Identifiers:

A name given to class, method or function

Rules of Identifiers:

1) Identifiers should always start with alphabet.
2) It can't start with digit.
3) It can have n number of digits after alphabet.
4) Identifiers are case sensitive and keywords cannot be identifiers.
5) Only $ and _ special characters are allowed

When class is declared as public you should declare classname as same as name of file name.

Variables:

A named memory location.

JDK:

It will provide platform for developing applications.

JRE:

It is more than enough for just running java programs.

Datatype:

It will specify the type of data that we want to store in a variable.

Java is strongly typed programming language.

2 types of Datatypes:

1) Primitive Datatype
2) Reference Datatype

   8 Primitive Datatype:

   1) Byte: 1 byte -128 to 127
   2) Short: 2 bytes -32768 to 32767
   3) Int: 4 bytes -2 billion to 2 billion
   4) Long: 8 bytes -9 billion to 9 billion
   5) Float: 4 bytes
   6) Double: 8 bytes
   7) Char : 2 bytes (0-65536)
   8) Boolean: 1 byte

Declaration:

Datatype var;

Initialization:

Var = value;

System.out.println (var)//utilization

Operators:

An operator is a special symbol or keyword that is used to designate mathematical operation or some other type of operation.

These operations can be performed on one or more than one values called as operands.

1) Arithmetic Operators: + ,- , *,/,%
   +, - : Additive arithmetic Operators
   *, / , % :Multiplicative Arithmetic Operators
   Output of Arithmetic Operator is integer
   Which ever the value of variable is greater than it will take data type of higher datatype
2) Assignment Operator
3) Increment/ Decrement Operator:
   ++,--(Unary Operator)
   A++ ->Post Increment
   ++a -> Pre Increment
   a-- ->Post Increment
   --a - > Pre Decrement

   Pre Increment
   1) Increment
   2) Substitute
   3) Utilize

   Post Increment
   1) Substitute
   2) Utilize
   3) Increment
4) Relational Operator:
   Output is boolean
   = = , !=, <, >, >=, <=
   - = = It returns true when both the side of equation are equal else false.
   - ! = It return true when both the side of equation are not equal else false

- < It returns true if left side of the equation is less than RHS
- > It returns true when left is greater than right.
- >= It returns true when left is greater than or equal right.
- <= It returns true if left side of the equation is less than RHS

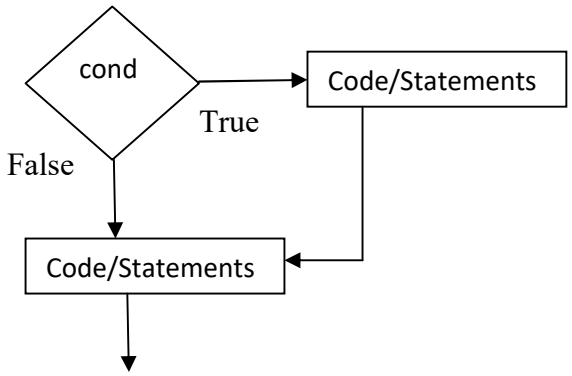
5) Logical Operators:
   Output is boolean

   Whenever we need to compare more than two conditions we use Logical Operators.
   !, &&, ||
   - ! Unary operator: It will return false if the right hand side of the operator is true.
   - && Binary Operator
   - || Binary Operator

6) Bitwise Operators:
   &, |
   Result of Bitwise operation is Integer

FLOW CONTROL STATEMENTS

1) If:
   Syntax:
   If(condition){
   Statements….
   statements…
   }



Arrays:

It is group of homogenous data with some index.

Index of an array will always starts from zero.

It has fixed size.

MaxIndex is size-1

Syntax:

Declaration of Array

datatype [ ] arrayname;

datatype[ ] arrayname;

datatype arrayname[ ]; //mostly not used

Creation of Array:

Int arr[] =new int[SIZE];

Initialization of Array:

Int arr[index] = value;

Array Index Out of Bound Exception:

If we try to access element from an array more than the specified size.

Length: if we length variable is used with any of the arrays it will give the size of the array.

Strings:

String is a sequence of character.

Strings in java are Immutable.

String is a userdefined datatypes.

Internally string is storing character array to store the data.

Syntax:

String str = “Hello”;

Or

String str;

Str = “Hello”;

Or

String str = new String(“Hello”);

String Methods:

1) Length():
   Rerurn type integer
2) tocharArray():
   Return type is character array.
   Converts whole string into character array.
   Length of character array = length of string
   First letter in character is equal to first letter in string.
   Array Index Out of Bound Exception.
3) charAt( ) : charAt(index)
4) equals( ):not null
   should be string
   character sequence should be same
5) equalsIgnorecase( ):
6) contains( ):
7) replace( ): str.replace(oldchar, newchar);
   old value is replaced by new value
8) indexOf() :return index of first occurrence

Polymorphism

When single entity acts differently for different objects it is known as polymorphism.

- Method overriding is a type of polymorphism. It is runtime polymorphism.
- Method Overloading is also a type of polymorphism. It is Compile time polymorphism.
- Run time polymorphism : Binding of method declaration and definition is done at run time. It is also known as Late Binding.
- Compile time Polymorphism : Binding of method declaration and definition is done at compile time. It is known as Early Binding.

Abstraction

Each and every invocation in java is abstraction.

The implementation details of the method are not taken into consideration, only the actual functionality.

Definition:

Hiding the implementation details by just providing the functionality.

Abstract class:

- Any class that has been defined using keyword abstract is known as Abstract Class.
- Any method that has been declared with abstract keyword is known as Abstract Method.
- Abstract method does not have body.
- Abstract classes can have both abstract methods as well as concrete methods.
- Any class having an abstract method should be declared as abstract. But vice-versa is not possible.
- If a class has been declared as abstract, then that class should be extended by sub class or child class.
- Implementation of abstract method is given in child class.
- The objects can't be created for abstract class.
- Abstract classes can have constructor.
- For abstract classes we can achieve zero to 100% abstraction.

Note:

- ⇨ It become optional to implement all the abstract methods of abstract class if we declare the child class also as a abstract class.
- ⇨ If a child class is not abstract then, we have to implement all the methods of super class.
- ⇨ If abstract class is implementing abstract class, then there is no need to implement all the methods.

Interface:

It is just a abstract class.

- A class can extend a class but it can only implement the interface.
- Any method inside interface is by default abstract.
- The data members are by default static, final .
- Interfaces do not have constructors.

Difference between Interface and Abstract class

INTERFACE	ABSTRACT CLASS
Interface can have only abstract methods Concrete methods can be present but declare the methods with static or default keyword. By default concrete methods are static.	Abstract class can have abstract and concrete Methods.
An interface can extend another Java interface only. One interface can extend any number of interface	An abstract class can extend another Java class and implement multiple Java interfaces.
interface can be implemented using keyword "implements"	Abstract class can be extended using keyword "extends".
Interface has only static, public and final variables	Abstract class can have final, non-final, static and non-static variables.
Members of a Java interface are public by default	A Java abstract class can have class members like private, protected, etc.
Not Possible	Abstract class can provide the implementation of interface

Functional Interface:

Interface which has only one abstract method is called Functional Interface.

@FunctionalInterface is annotation is used to specify or ensure that it is functional Interface.

Syntax:

@FunctionalInterface

Public interface interfacename{

……

}

1)Marker Interface :Abstract method should not be present inside it. It is used to provide special behavior to class

Ex: Serializable is marker interface and Remote is also marker interaface.

2)Functional Interface :One abstract method

3)Normal Interface: N number of abstract methods and concrete methods.

Note :

Packages other than Java.lang should be imported in the class file.

Encapsulation:

Hiding the datamembers is called Encapsulation.

- Getters: public methods which are used to access the private data members from the class.
- Return type is dependent on datamember
-

Setteres :

- Used to manipulate the private datamembers in the class.
- Return type is void

⇨ Encapsulation is a mechanism with which we wrap up the data (data members and function members) into single object.

⇨ Encapsulation is used for data hiding.

⇨ If data is read only ->getters methods

⇨ If Data is write only->setters methods

PACKAGES

Java API is the set of Packages ex:java.lang

If we want to access a class name which is same that is present in two different we have to import one package and create object using fully qualified class name for other class name that we want.

Final

Final variables cannot be changed once declared.

Final variables are created using block letters

Final methods cannot be reinitialized and final mehods cannot be overridden.

Final classes cannot be inherited.

Final classes can inherit from super classes.

Object Class

In java each and every class directly or indirectly inherits the properties of Object class i.e.,(in essence of)

Object class is the super most class in java. Each and Every class either a predefined class or user defined class is child class of Object.

Each and every class directly or indirectly inherits from Object Class.

Methods of Object Classes:

1. getClasses( ): return type is class
2. clone()
3. equals()
4. toString()
5. wait() :Used in threads
6. wait(long)
7. wait(long, int)
8. notify()
9. notifyAll()
10. finalize() :used with garbage collection. It is invoked as soon as the execution of program ends
11. hashCode(): return type is integer
    It gives address of memory location of object
    If we override hashcode and equals method, the two objects will be same otherwise they will be different

Collection Framework

1. Set Interface

Allows null elements

2. List Interface:

3. Queue Interface:

FIFO

Boolean Add(E,e): inserts the specified elements to queue if it is possible to do so immediatey without violating capcity restrictions

It throws following Exception

1) ClassCastException
2) NullPointerException
3) IllegalArguementException

Offer()inserts specified element to queue but should not violate capacity

It throws following Exception

1) ClassCastException
2) NullPointerException

3) IllegalArguementException

Boolean Offer(E,e) : returns true if the element was added to queue:

Use inside try block

i) Classnotfound exception
ii) Null pointer exception
iii) Illegalarguement exception

Remove() :Retrieves and removes the head of this queue. Or returns null if queue is empty

Poll() : it retrieves but not remove the head of this queue.

If queue is empty it will throw NoSuchElementException

Element(): It retrieves but does not remove the head of this queue

It returns exeception object.

Peek() : returns head of queue if queue is empty.

Peek will not will give exeception.

Map interface

Map cannot contain duplicate value

It cannot contain duplicate keys

Each key can map to atmost one value

Java map is an object that maps keys and values

It consists of 3 general purpose map implementation:

1) Hash Map
2) Tree Map
3) 1Linked Hash

Put,get,containsKey, size, is Empty, containsValue

Map is the only interface that does not inherit from collection interface

Java Collection Classes

1) Hashset Class : It have 5 Constructors

⇨ Implementing set interface

- ⇨ Permits null elements
- ⇨ No iteration order of the set and permits null element
- ⇨ Add,remove,contains and read methods
- ⇨ We can set initial capacity and load factor
- ⇨ Load factor: measure of amout of element inside a set.
- ⇨ This class implements the supported set interface backed by hash table.
- ⇨ It makes no guarentees as to the iteration order of the set
- ⇨ This class permits null element
- ⇨ 1)Public HashSet(){// These are the constructors
  Map = new HashMap<>();

  }
- ⇨ Initial capacity =>16 Load Factor : 0.75
- ⇨ 2) public HashSet(int initialCapacity, float loadfactor){
  Map = new HashMap< >(initial capacity,loadfactor);
  }
  We can change load factor using this constructor
- ⇨ 3) linked Hash map
  HashSet(int initialcapacity, float loadfactor, Boolean dummy){
  Map = new LinkedHashMap<>(initialcapacity, Loadfactor);
  Methods:
  1) Add()
  2) Remove()
  3) Clear()
  4) Clone( ): return type is object

HashMap:

Implements map intrerface

It does not implement collections

1) Put( ) output is empty for the very first time.
2)