**Java**

Java is a class-based object-oriented programming language.

* Platform independent

Once compilation is done it is made into byte code so that it can run in any platform.

OOPS Concept in Java

1) Class

2) Object

3) Inheritance

4) Polymorphism

5) Abstraction

6) Encapsulation

Variables

* Used for storing data
* Declare a variable - data\_type variable\_name

**Data types**

* byte – 1byte
* short - 2bytes
* int – 4 bytes
* long – 8bytes
* float - fractional numbers with 6 decimal digits (4 bytes)
* double – fractional numbers with 15 decimal digits (8 bytes)
* boolean – true or false values(1 bit only)
* char – single character(2 bytes) for eg a , b etc
* Strings – Store texts(more than one character) for eg: amma , ammu etc

Decision making statements

if Statement:

* An if statement consists of a Boolean expression followed by one or more statements.

Syntax:

if(Boolean\_expression) {

//Statements will execute if the Boolean expression is true

}

if...else Statement:

An if statement can be followed by an optional *else* statement, which executes when the Boolean expression is false.

Syntax:

if(Boolean\_expression){

//Executes when the Boolean expression is true }

else{

//Executes when the Boolean expression is false }

if...else if...else Statement:

An if statement can be followed by an optional *else if...else* statement, which is very useful to test various conditions using single if...else if statement.

**In multiple if statements it is possible to have more than one condition true. but in the case of if elseif else it evaluates the 2nd condition only when the 1st condition is false**

Syntax:

The syntax of an if...else is:

if(Boolean\_expression1){

//Executes when the Boolean expression 1 is true }

elseif(Boolean\_expression2){

//Executes when the Boolean expression 2 is true

}elseif(Boolean\_expression3){

//Executes when the Boolean expression 3 is true}

else {

//Executes when the **none of the above condition** is true.}

Nested if...else Statement:

use one if or else if statement inside another if or else if statement.

Syntax:

if(Boolean\_expression1){

//Executes when the Boolean expression 1 is true

if(Boolean\_expression2){

//Executes when the Boolean expression 2 is true }

}}

Switch

* Similar to Multiple if-else statement.
* It is a keyword and we have conditions in it.
* Value of a case should be a constant.
* There can’t be any duplicate case value.
* We can put break statement in between case, if break statement is not there it continues the execution.
* Condition inside the switch can be of any primitive type/enums/Strings.
* It is not compulsory to have default statement in a switch.

Syntax:

switch(expression){

**If – to specify a block of code to be executed, if a specified conditions is true.**

**Else – to specify block of code to be executed, if the condition is false.**

**Else if – to specify a new condition to test, if the 1st condition is false.**

**Switch – to specify many alternative blocks of code to be executed**.

The break statement is used to terminate the **switch-case** statement.

If break is not used, all the cases after the matching case are also executed.

case value :

//Statements

break;//optional

case value : //Statements

break;//optional

//You can have any number of case statements.

default://Optional

//Statements

}

Looping statements in Java

Used to iterate a part of the program several times

For loop

* iterate a part of the program several times
* if the number of iteration is fixed it is recommended to use for loop
* simple for loop, nested for loop, for-each loop, labelled for loop, infinite for loop.

SYNTAX

for (initialization; condition; increment or decrement)

{

// code to be executed

}

Enhanced for loop in Java:

* This is mainly used for Arrays.

Syntax:

for(declaration : expression) {

//Statements

}

While loop

* it is control flow statement that executes the prgm repeatedly based on given Boolean statement
* used when the number of iteration is not fixed.

SYNTAX

while (condition)

{

// code to be executed

}

//While loop

Int i=1;

while(i<=10)

{

System.out.println(i);

i++;

}

do- while loop

* it is control flow statement that executes the prgm atleast once and further execution is based on given Boolean statement.
* Used when the number of iteration is not fixed and we have to execute the program atleast once
* If you pass true in the do-while loop, it will be infinite do-while loop.

SYNTAX

do

{

//code to be executed

}

while(condition);

Block

* Set of code that is within the curly braces{} within any class, method or constructor.

Syntax

{

//block of code

}

break Keyword:

* breakkeyword is used to stop the entire loop.
* The break keyword must be used inside any loop or a switch statement
* The break keyword will stop the execution of the innermost loop and start executing the next line of code after the block.

Syntax:

break;

Array

* Can hold elements of same datatype
* Elements are stored in contiguous memory location.
* Arrays work with respect to Index
* Start with index 0
* Size limit - We can store only fixed size of elements in an array
* Random access- can get any data located at an index position.
* There are 1d as well as multidimensional arrays.

SYNTAX

datatype arr[]

for eg :- int a[] = {1,2,3,4,5,7};

**Constructors**

* Used to create an object.
* By default any class will be supplied with ZERO argument constructor.
* Once a constructor is supplied then ZERO argument constructor will not be there.
* It is named so becoz it constructs the value at the time of object creation.
* Every time an object is created using new() keyword , atleast 1 constructor is called.

( it is called with an instance of class is created(object))

* At the time of calling constructor memory for object is allocated in memory.
* Name of constructor should be same as its class name.
* Java constructor cannot be abstract, synchronized, final and static.
* We can use access modifiers (private, protected, public or default) while declaring a constructor.
* 2 types - no arg constructor(default constructor)

- Parameterized constructor

**Static**

* Not an instance variable
* It’s a class variable or global variable
* Used to save the memory
* We can apply static to variables, classes, methods and blocks.

**This**

* it is a reference variable which points to the current object.
* this can be used on instance variable, passed as an argument, and with the constructor also.
* If local variables and instance variables are same then we can use “this” to distinguish btwn them.
* Invoke current class constructor
* Invoke current class method
* Return the current class object
* Pass an argument in the method call
* Pass an argument in the constructor call

**Super**

* It is a reference variable which points to the parent class object
* It can be used with methods, constructors and variables

**Method Overloading**

* Return type doesn’t matter
* If a class has multiple methods having same name but diff parameters or different datatypes then it is known as Method Overloading
* 2 types - By changing the datatype

By changing the arguments

* Not possible by changing the return type of method only.

**Method Overriding**

* If subclass has same name method as declared in parent class
* Method must have same name as in the parent class
* Method must have same parameter as in the parent class

**Exceptions**

* Problem arises during the execution of the program.

Reasons – entered invalid data, file need to be opened cannot be find, network connection has been lost in the middle of communication or JVM has run out of memory

Categories of exception

* Checked exceptions – User error

**Cannot** simply ignored during compilation.

* Runtime exceptions – **Can** ignored during compilation.
* Errors – Not an exception at all

**Can** ignored during compilation.

Ignored in code

Eg : if a stack overflow occurs, an error will arise

* **BufferedReader** – reads a couple of characters from the input Streams and store that in buffer
* **InputStreamReader** -reads only one character from the

**Collections in java**

* Collections is a class in java.
* Framework that provides an architecture to store and manipulate the grp of objects
* It can contain heterogenous elements and can modify the size
* Collection means a single unit of objects.
* It provides many
* Collection is an interface
* List ->arraylist, Linkedlist
* Map -> HashMap
* Set -> TreeSet, HashSet
* Queue -> PriorityQueue

**Threads in java**

2 ways to create threads in java

* By extending thread class
* By implementing runnable interface

**Extra points**

box obj = **new** box();

System.out.println(obj);

}

}

Answer:  classname@hashcode in hexadecimal form

Explanation: When we print object internally toString() will be called to return string into this format classname@hashcode in hexadecimal form.

* Return type of a method must be made void if it is not returning any value.
* A constructor is a method that initializes an object immediately upon creation. It has the same name as that of class in which it resides.
* main() method can be defined only once in a program. Program execution begins from the main() method by java runtime system.
* All object of class share a single copy of methods defined in a class, Methods are allotted memory only once. All the objects of the class have access to methods of that class are allotted memory only for the variables not for the methods.
* Constructors does not have any return type, not even void.
* this keyword can be used inside any method to refer to the current object. this is always a reference to the object on which the method was invoked.
* finalize() - function is used to perform some action when the object is to be destroyed. finalize() method is called just prior to garbage collection. it is not called when object goes out of scope.
* If parent class constructor throws any checked exception, compulsory child class constructor should throw the same checked exception as its parent, otherwise code won’t compile.
* Private constructor ensures only one instance of a class exist at any point of time. Object of private constructor can only be created within class. Private constructor is used in singleton pattern.
* this() and super() cannot be used in a method. This throws compile time error.
* “this” and “super” can be used in a constructor
* Class.getInstance creates object if class does not have any constructor
* constructor can take any number of parameters