Core Java

Q. What is Java?

Ans:

1) Java is a general purpose language.
2) It means Java language can be used to develop different types of applications.
3) In other words Java is not domain specific so it can be used in various fields.

Q. Explain. The different versions of Java programming language.

Ans:

1) Java programming language has 3 different versions:
    a) JSE(Java Standard Edition)
        i) JSE is also known as core Java.
        ii) JSE can be used to develop desktop applications.
    b) J2EE(Java Enterprise Edition):
        i) J2EE is also known as advanced Java.
        ii) J2EE can be used to develop web applications.
    c) JME(Java Micro Edition):
        i) JME is also known as the Android edition.
        ii) JME can be used for the development of an android application.

Q. What are the application areas of Java language?

Ans:

1) Java can be used in the following fields.
   a) Desktop application development.
   b) Enterprise application development.
   c) Android application development.
   d) Automation tools.
   e) Big data technologies.

Q. Explain the features of Java language.

Ans:

1) Simple
   a) Java is a simple programming language because we can easily create and maintain Java syntax.
   b) It is possible to learn Java without having knowledge of other programming languages.
2) Platform independent
   a) Java is a platform independent language; it means we can execute Java programs by using any OS platform.
3) Portable
   a) Java is a portable programming language. It means we can switch the development environment from one operating system to another operating System
   b) In other words we can develop Java applications by using various platforms.

4) Object-oriented
    a) Java is an object oriented programming language because we can represent each and everything in terms of objects.
5) Multithreaded
    a) Java is a multi-threaded programming language which means we can develop multi-threaded programs using Java. ... Each of the threads can run in parallel.

Q. Explain the development process of java application?

Ans:

1) Coding
    a) The process of writing Java statements inside the .Java file is known as coding.
    b) The file which contains the source code is also known as source file.
2) Compilation
    a) After coding we have to compile the .java file. Java compiler is responsible for compilation.
    b) The Java compiler performs the following activities in order to complete the compilation process.
        i) Checks the syntax and rules of language.
        ii) Converts Java statements into the byte code.
    c) After successful compilation the compiler will generate . Class file.

3) Execution
    a) JVM is responsible for the execution of the .class file.
    b) To complete the execution process we have to install the JDK software.

Q. Explain the different environments present in the IT industry.

Ans:

1) Following are the environments present inside the IT industry.
    a) Development environment.
        i) Developers are responsible for coding, compilation and execution processes. In the development environment we have to install JDK software.
    b) Testing environment.
        i) Testers are responsible for the execution of software applications. In a testing environment we have to use only JRE.
    c) Production environment.
        i) In production environments beta testing will take place. We have to install only JRE in the production environment.

Q. What is JDK?

Ans:

1) JDK stands for Java development kit. It is the collection of development related
2) components, for example: compiler, JVM, JIT. JDK is an installable software for the Java programming language.

Q. What is JVM?

Ans:

1) JVM stands for Java virtual machine. JVM is responsible for the interpretation of byte code. In other words JVM is the component which understands the meaning of code.
2) JVM is responsible for the execution of class files.

Q. What is JIT?

Ans:

1) JIT stands for Just In Time compiler. JIT is responsible for converting byte code into machine code in order to improve the performance of Java programs.

Q. What is JRE?

Ans:

1) JRE stands for Java Runtime Environment.
2) JRE provides an execution environment for JVM in order to execute .class files. Without JRE JVM cannot execute the class file.

Q. Explain. Configuration process of JDK.

Ans:

1) JDK is an installable software for the Java programming language. After installation it is mandatory to configure Java components.
2) In order to configure Java components we have to create path variables by using advanced system settings.

Q. What is a path variable?

Ans:

1) Path variable represents location of different Java components. We have to provide the location of the bin folder present inside the JDK as below for the path variable.
2) Following are the steps to set path variable:
   a) Go to C:-> Program Files/ Java/ JDK/ bin. Copy this path.
   b) Go to this PC properties-> advanced system settings-> environment variables.
   c) Create a new variable as path and paste the copied path as value for that variable.

Q. Explain the structure of the Java program.

Ans:

```
class class_name
{
        public static void main (string[] args)
        {
        Java statements;
        }
}
```

Q. What is the use of + operator?

Ans:

1) + Operator can be used to perform addition and concatenation.
2) We can pass different types of input for + operator.
3) If the expression contains only numeric values, only character values or else numeric and character values then addition will take place.
4) If the expression contains string value JVM will perform concatenation.
5) Concatenation is the process of joining two different inputs (concatenation is applicable only for string input).

Q. What is a data type?

Ans:

1) Data type is used for storing a specific value of type.
2) Data types can be used to refer to different types of values.

3) In Java data types are classified into 2 types
    a) Primitive
    b) non-primitive

Q. Explain primitive data types available in Java.

Ans:

1) Java language has provided 8 primitive data types:
    a) int : 4 bytes.
    b) float: 4 bytes.
    c) double: 8 bytes.
    d) char : 2 bytes
    e) long(int): 8 bytes.
    f) short(int): 2 bytes.
    g) boolean: 1 bit
    h) byte: 8 bits.

Q how to store string values in Java?

Ans:

1) We have to use a predefined class named String.
2) String is not a primitive data type, so it is considered as a non-primitive data type.

Q. What is a variable? Explain the usage of variables.

Ans:

1) Variables are used to store program data.
2) Variables act as a container for different types of program information.
3) To store the data in terms of variables, we have to follow 3 steps:
    a) Declaration
    b) Initialisation.
    c) Re-initialization.
4) The process of creating a variable by using an appropriate data type is known as declaration.
    a) Data type variable name;
5) The process of assigning the initial value to an already declared variable is known as initialization.
    a) Variable name= value;
6) The process of assigning different values to the already initialised variable is known as re-initialization.
    a) Variable name= value;

Q. Explain the difference between keywords and identifiers.

Ans:

1) Keywords:
    a) All the keywords are predefined words.
    b) The words provided by Java language are known as keywords.
    c) Every keyword in Java has a specific meaning.

    d) All the keywords are reserved words because we cannot use keywords as a variable name, class name or method name.
    e) It is mandatory to declare all the keywords in lowercase format. For example:
    f) class, public, static, int, etc are keywords.
2) Identifiers:
    a) Identifiers are the words created by developers.
    b) Variable name, class name and method name will be considered as identifiers.
    c) To create an identifier we have to follow some rules: 1. Every identifier must and should contain alpha or alpha-numeric.
    d) Identifier names should start with the alphabet or else ‘_’.
    e) For example: ABC123 is valid.
    f) 123abc is invalid.

Q. What are the industry naming standards?

Ans:

1) Every class name, interface name should start with uppercase letter and every subsequent word should also start with uppercase letter.
    a) class SavingAccount
2) Every variable name, method name should start with lowercase letter and every subsequent word should start with uppercase letter.
    a) int accountNumber;

Q. What is the use of method in Java programming?

Ans:

1) Method is a collection of executable Java statements.
2) Java methods can be used to develop business logic as well as to perform operations on program data.
3) The main advantage of using a method is reusability, it means "Write once, execute multiple times".

Q. Explain the types of methods.

Ans:

1) In Java methods are mainly classified into two types:
    a) Internal method:
        i) Methods provided by the Java language are known as internal methods.
        ii) JVM is responsible for the execution of internal methods.
        iii) It is not possible to modify the signature of an internal method.
    b) External methods:
        i) Methods created by developers are known as external methods.
        ii) Developer is responsible for the execution of external methods.
        iii) It is mandatory to call the external method inside the main method.
        iv) Developers can create any number of external methods.

Q. Explain the return type of method.

Ans:

1) Every method in Java must have a return type.
2) If the return type is void it means the method cannot return any value.
3) We can transfer the data from one method to another method but using a return statement.
4) If the method returns the value then we have to change the return type of a specific method.
5) Return type should be similar to the data type of value which is going to be returned by method.
6) At a time method can return only one value, because we cannot use multiple return statements.
7) Multiple return statements are not allowed because a method should have only one return type.

Q. What is the use of the Scanner class?

Ans:

1) Scanner class is an internal class which can be used to accept input from the end user.
2) We can accept different types of input by using the Scanner class.
3) To Accept the input from user we have to follow 3 steps
    a) We have to import the scanner class from the java.util package.
        i) Import java.util.Scanner;

b) After importing the package, we have to create a reference of scanner
    i) Scanner s = new Scanner(System.in);
c) After creating a reference of scanner class we have to call internal
d) methods of scanner class to different types of input.
    i) next() : For String and character input.
    ii) nextInt(): for Integer input.
    iii) nextfloat(): for decimal
    iv) nextdouble(): for Decimal.
    v) nextLong(): for long int
    vi) nextShort(): for short int
    vii) nextBoolean(): for Boolean input;
    viii) nextByte(): for Byte input;
    ix) nextLine(): to accept multiple strings in single line

Q. What is the use of control statements?

Ans:

1) Control statements can be used to apply conditions or restrictions on Java statements.
2) Developers can control the flow of execution of a program by using different control statements. In Java control statements are classified into 2 categories.
    a) Decision making statement.
    b) Looping statement.

Q. What is meant by a decision making statement?

Ans:

1) Decision making, it means programmers can decide execution of Java statements based on specific conditions.
2) In Java "if statement" is used for decision making operations.

Q. Explain the working of if and else statements?

Ans:

1) If statements always accept conditions from the developer, if the condition is true then respective Java statements will be executed.
2) If statement always returns Boolean type of value i.e true or false.

   Following are the basic scenarios of if else statement:

```
if(condition)
{
Java statements;
}
==============================
if(condition)
{
Java statements;
}
else{
Java statements;
}
===============================
if(condition)
{
Java statements;
}
else if(condition)
{
Java statements;
}
.
.
.
else
{
Java statements;
}
```

Q. Explain the concept of members.

Ans:

1) Anything declared inside the class body will be considered as members of a class.
2) Members are mainly classified into two types:
    a) data member
    b) function member.
3) Data members are nothing but variables declared inside the class.
4) Methods of the class will be considered as function members.

5) Members of the class body can be static or non-static.
6) MEMBERS
    a) DATA MEMBERS:
        i) Static variable.
        ii) Non-static variables.
    b) FUNCTION MEMBERS:
        i) Static methods.
        ii) Non-static methods.

Q. Is it possible to create multiple classes inside the single .java file?

Ans:

1) Yes it is possible to create multiple classes inside a single .java file.
2) In such a case, only one class should contain the main method.
3) The class which contains the main method is known as main class or executable class.
4) The class which does not contain the main method will be considered as
5) business logic class or functional class.
6) Filename should be the same as a class name which contains the main method.

Q. Explain the static members of the class.

Ans:

1) Members declared along with static keywords are known as static members of the class.
2) Static means a common accessible resource (everyone can access the same).
3) Static members will be having a single copy into JVM memory, to access static members we have to provide a class name as reference.

Q. Explain non-static members of a class.

Ans:

1) Members declared without using static keywords are known as non-static members.
2) Non-static members will be having multiple copies inside the JVM memory.
3) To access non-static members outside the class or else inside the static context we have to create an object.
4) We can create an object by using a new operator.

Q. Explain accessibility rules for members of the class.

Ans:

1) Static context.
    a) Static members (without any reference inside the same class)

    b) Static members (with reference class name, outside the same class)
    c) Non-static members (with reference to an object, same & different class).

2) Non static context:
    a) Non static members (without any reference inside the same class).
    b) Non static members, with reference outside the class.
    c) Static members (same class= no reference required)

Q. Explain concept of reference variable.

Ans:

1) Reference variable is a class type variable, which holds the address of an object.
2) Reference variables can be used to access non-static members of a specific class.
3) Default value of every reference variable is always null.
4) If we print the reference variable, the address of an object will get printed.
5) We can create multiple reference variables for a single object.
6) It is possible to copy the address of one reference variable into another reference variable.

Q. What is an object?

Ans:

1) An entity having its own state and behaviour is known as an object.
2) State represents characteristics of an object, whereas behaviour represents functionality.
3) In terms of Java, an object is a copy of non-static data members and function members.
4) Non static data members represent the state of an object, whereas non static function
5) members represent behaviour of an object.
6) Class is a container for object state and behaviour, hence we have to create an object of a class.
7) In Java an object is also known as an instance of class.

Q. Explain use of Java operators.

Ans:

1) Operators are mainly used to perform operations on data.
2) Java operators play an important role in order to control program execution.
3) In Java, operators are classified into 6 categories:
    a) Arithmetic
    b) Assignment
    c) Logical
    d) Relational
    e) Unary

f) Bit-wise.

Q. Explain arithmetic operators.

Ans:

1) Arithmetic operators: Are responsible to perform arithmetic operations in Java programs.

Operator Name	Usage	Example
+	Addition and concatenation	c=a+b
-	Subtraction	c=a-b
*	Multiplication	c=a*b
/	Division	c=a/b
%	Modulo	c=a%b

Q. Explain assignment operators.

Ans:

Operator Name	Usage	Example
=	Assignment	c=20
+=	Addition & Assignment	a+=b
-=	Subtraction & Assignment	a-=b
*=	Multiplication & Assignment	a*=b
/=	Division & Assignment	a/=b

%=	Modulo and Assignment	a%=b

Q. Explain relational operators.

Operator Name	Usage	Example
==	Check for equality	a==b
!=	Check for inequality	c=a-b
<	Less than	a<b
<=	Less than equal	a<=b
>	Greater than	a>b
>=	Greater than equal	a>=b

Q. Explain Unary operators.

Ans:

1) Unary operators are used to perform the operation on a single variable. Unary operator classified in 2 categories
    a) Increment
    b) Decrement.
2) Post increment: it means first use the current value inside the expression then increment the original value.
3) Pre-increment: it means first increment the original value then use the latest value inside the expression.

4) Post decrement: it means first use the current value and decrease the original value after assignment.
5) Pre-decrement: it means first decrease the original value and then use inside the expression.

Q. Explain logical operators.

Ans:

1) Logical operator acts as a conjunction between multiple conditions.
2) We can use these operators to perform logical operations on program data.
3) Following are the operators available in this category.
    a) Logical and operator(&&)
    b) Logical or operator(||)
    c) Logical not operator(!)

&& operator

INPUT1	INPUT2	OUTPUT
T	T	T
T	F	F
F	-	F

|| operator

INPUT1	INPUT2	OUTPUT
T	-	T
F	T	T

F	F	F

- Logical not: this operator always gives the negation of given value. If you are using 'not' operator then output will always be Boolean value. We can use this operator along with other logical operators.

Q. Explain Bitwise operators.

Ans:

1) They are used to perform operations on individual bits.
2) They are applicable only for integer data types. For example: int short long byte.
3) It always converts a given decimal value in the binary format before performing operations.
4) It is classified in following types:
    a) Bitwise and: &
    b) Bitwise or: |
    c) Left shift: <<
    d) Right shift: >>

& operator

INPUT1	INPUT2	OUTPUT
1	1	1
1	0	0
0	0/1	0

| operator

INPUT1	INPUT2	OUTPUT
1	1/0	1
0	1	1
0	0	0

1) Bitwise left shift:
    a) left shift can be used to shift the left by the number of times specified by the users.
    b) In the case of the left shift operator decimal value will be converted into the binary format before performing shift operation.
    c) We can use one formula to calculate the value of the left shift expression.
    d) Shifting the digit by one means multiplying by 2.
    e) For example: int a = 10, int result= a<<3.
    f) 1st shift : 10*2: 20
    g) 2nd shift: 20*2: 40
    h) 3rd shift: 40*2: 80
2) Bitwise Right shift operator
    a) It can be used to shift the given value to the right side by the number of times specified by the user.
    b) In the case of the right shift operator as well, the given decimal value will be converted into binary format before performing any operation.

c) In order to Perform right shift operation we cannot exceed the limit of total number of positions. It means for right soft operation the end point is first position.

d) If we are exceeding the limit then output will be 0.

Q. Explain Ternary Operator.

Ans:

1) It can be used to apply conditions and expressions in a Java statement.
2) It is an alternative to if-else statements.
3) This operator accepts or operates on 3 parameters, hence it is known as ternary operator.
4) Syntax: var_name=(exp1...n)? Exec 1:exe2;

Q. Explain JVM memory architecture.

Ans:

1) JVM memory architecture is a collection of different components.
2) JVM memory architecture classified into 3 sections:
    a) Class loader subsystem
    b) Runtime memory
    c) Execution engine.
3) Class loader subsystem is responsible to perform following activities:
    a) Loading
    b) Linking
    c) Initialisation.

4) Loading
    a) In this phase Java classes will be loaded inside the JVM memory. In Java 3 types of class loader are present
        i) Bootstrap class loader.
        ii) Extension class loader.
        iii) Application class loader.
    b) Bootstrap class loader is responsible to load inbuilt class inside the JVM memory for example scanner, string,etc
    c) Extension class loader is responsible for the loading of external classes or 3rd party classes. For example database classes, file classes, server classes, etc.
    d) Application class loader this class loader is responsible to load the classes created by developer (use defined classes)
5) Linking: in this phase JVM will perform three tasks in order to link the byte code:
    a) Verify: in this phase JVM will verify the byte code generated by the compiler. If there is a problem in byte code then JVM will throw a run-time error.
    b) Prepare: after verification default values will be assigned to all the class variables.
    c) Resolve: in this phase JVM will resolve internal issues related with byte code.
6) Initialisation: after linking all the static variables will be initialised with actual values.
7) Runtime memory: it is classified into five sections:

   a) Static pool area: this area acts as a container for all the static data members and declaration of static function members.
   b) Heap memory: In this section all the non-static data members and declaration of non-static function members. Objects of the class will be present inside the heap memory.
   c) Method area: this area acts as a container for all the static and non-static destinations of the method.
   d) PC registers: This section acts as a container for currently executing Java statements.
   e) Stack: is responsible to make a call for all the data members and function members.
8) In Java JVM itself acts as an interpreter, interpreter is responsible to understand the meaning of code.
9) JIT just in time the compiler is responsible to convert byte code into Machine code, in order to improve performance of Java applications.
10) Garbage collector: it deallocates the memory allocated for all the members of class after successful execution. It can be done implicitly as well as explicitly.

Q. Explain the process of class loading.

Ans:

1) The process of loading class into the JVM memory is known as class loading.
2) Class loading can be done implicitly and explicitly.

3) If a Java file contains only the main class then JVM will automatically load the respective class inside the memory.
4) In case a Java file is having more than one class then JVM is responsible for loading only the main class.
5) Programmer is responsible for loading the business logic class inside the JVM memory.
6) If a programmer is loading the business logic class then it is known as an explicit class loading.
7) We can load the business logic class by accessing any one member of the respective class.

Q. What is a block?

Ans:

1) Block is a collection of executable Java statements.
2) Blocks can be used to perform the special operations at the time of software development.
3) Blocks are mainly classified into 2 types
    a) Static
    b) Non-static block

Q. Explain the use of static blocks.

Ans:

1) The block which is declared along with a static keyboard is known as a static block.

2) Static blocks can be used to develop a business logic which is common for all the users, and will be executed only once in the application cycle.
3) If Java class is having static block and Main method then static block will execute before main method.
4) We can provide multiple static blocks in a single Java class, in such a case all the static blocks will be executed in a sequential way.
5) Static block executes before main method whereas main method will be executed after the class loading.

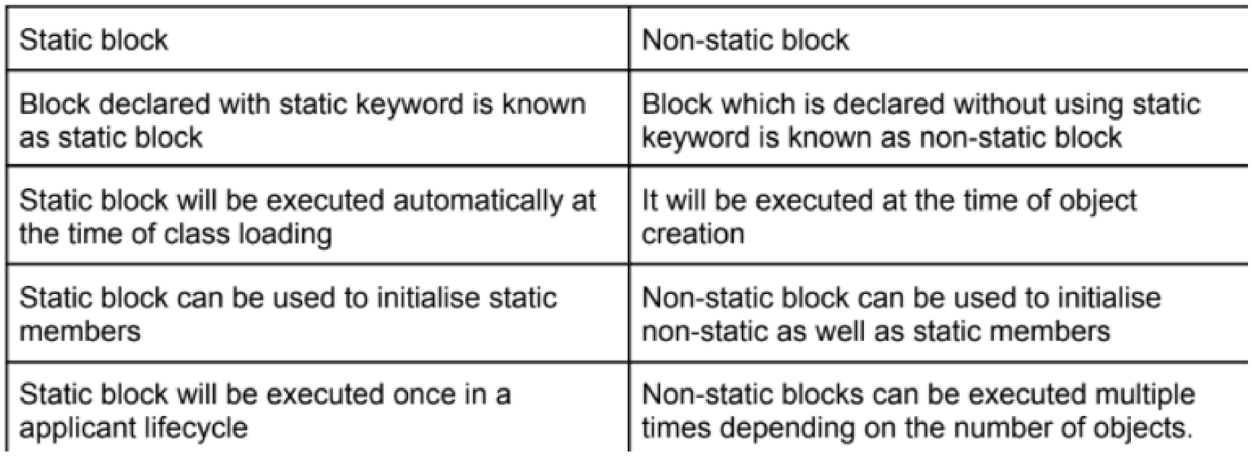
Q. What is a non-static block?

Ans:

1) Block which is declared without any keyword is known as a non-static block.
2) Non-static block is used to develop a business logic which is common for all the objects and will be executed after every object creation.
3) If we are creating multiple objects of the same class then multiple non-static blocks will be executed.
4) We can initialise non-static members inside the non-static block.

Q. Explain the difference between static and non-static block

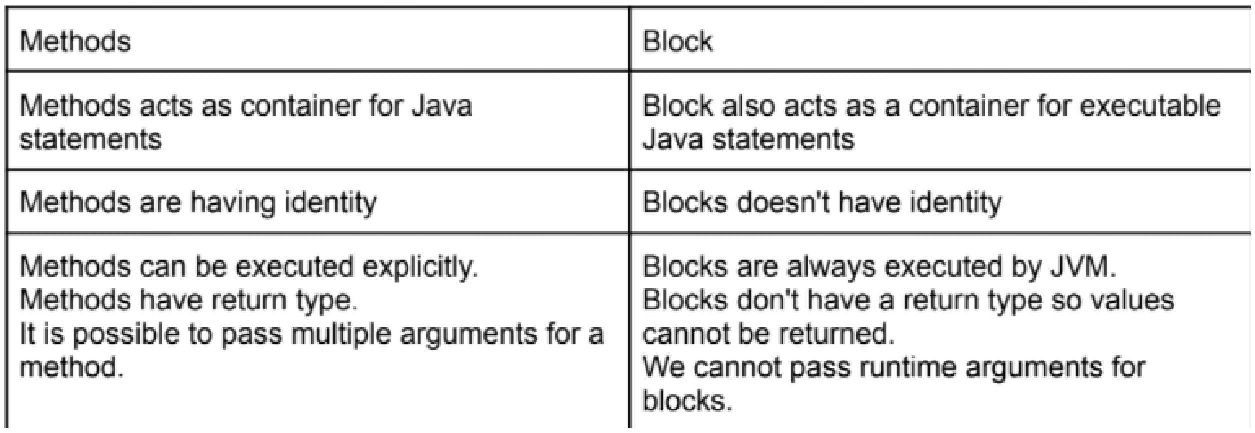
Ans:



Static block	Non-static block
Block declared with static keyword is known as static block	Block which is declared without using static keyword is known as non-static block
Static block will be executed automatically at the time of class loading	It will be executed at the time of object creation
Static block can be used to initialise static members	Non-static block can be used to initialise non-static as well as static members
Static block will be executed once in a applicant lifecycle	Non-static blocks can be executed multiple times depending on the number of objects.

Q. Explain the difference between blocks and methods.

Ans:



Methods	Block
Methods acts as container for Java statements	Block also acts as a container for executable Java statements
Methods are having identity	Blocks doesn't have identity
Methods can be executed explicitly. Methods have return type. It is possible to pass multiple arguments for a method.	Blocks are always executed by JVM. Blocks don't have a return type so values cannot be returned. We cannot pass runtime arguments for blocks.

Q. What is a constructor?

Ans:

1) Construct is a special member of a class which can be used to initialise the state of an object.
2) Class name and constructor name should be the same.
3) It is not possible to provide return type for a constructor.

4) Every Java class must and should have a constructor, either defined by the compiler or another programmer.
5) If the compiler defines the constructor then it is known as default constructor.
6) Compiler can define only zero argument constructor.
7) Compiler will provide a constructor only if the user defined constructor is not available.
8) Programmers can define zero arguments as well as augmented constructor.
9) Constructor will be executed at the time of object creation.

Q. How can a constructor return the address of an object?

Ans:

1) Developers cannot return any explicit value from the constructor because the constructor doesn't have a return type.
2) At the time of object creation a new operator is responsible to call the constructor of a specific class.
3) When the constructor call takes place automatically the constructor will return the address of an object.
4) It means it is not possible to return the address explicitly but by default constructor only return the address without support of return statement or return type.

Q. Why has Java provided 2 types of constructor?

Ans:

1) Construct calls will happen at the time of object creation.
2) We can create objects for every Java class.
3) If the default constructor is not provided by Java then every time the developer has to provide an external constructor.
4) User defined Constructor is mainly used to pass the data at the time of object creation.
5) If the default constructor option is not there then we have to create an external constructor even if non static members are not available. Because of all these reasons two types of constructors are available.

Q. What is a constructor overloading?

Ans:

1) The process of creating multiple constructors of the same class with different arguments is known as constructor overloading.
2) At the time of constructor overloading we have to pass different types of arguments for every constructor, agreements should be different either by augmented length or type.
3) Constructor overloading can be used to create different objects of the same class.

Q. What are the users of this keyword?

Ans:

1) This keyword always points to the current object.
2) It means address of an object and this keyword will be the same.
3) This keyword is mainly used to differentiate local and instance variables.

4) We can use this keyword only inside the non-static context, for example: constructor or non-static method.

Q. What is an array?

Ans:

1) ARRAY is a collection of multiple elements.
2) ARRAY can be used to store a group of the same type of elements.
3) It means we can store only homogeneous values inside the array variable.
4) In Java array classified into 2 types:
    a) Primitive array.
    b) Non-primitive array.(2D & 3D)

Q. What is a primitive array?

Ans:

1) An ARRAY variable created using primitive data type is known as primitive array.
2) Primitive array is also known as a data type array.
3) Primitive array further classified into 2 types:
    a) One dimensional array.
    b) Two dimensional array.

Q. Explain String class.

Ans:

1) String class is an internal class declared inside the java.lang package.

2) String class can be used to declare string values.
3) Every string value will be considered as a String object.
4) We can create String variables by using 2 ways:
    a) Without using a new operator.
    b) With using new operators.
5) All String values will be stored in the string pool area situated inside heap memory.
6) String pool area classified into 2 sections:
    a) Constant Pool area
    b) Non-Constant Pool area.
7) String Variables created without a new operator will get stored inside the constant pool area.
8) String variables created using the new operator will get stored inside the non-constant pool area.
9) Constant pool area does not allow duplicate values.
10) Non-constant pool area allows duplicate values.

What is the difference between == operator and equals method?

Ans:

1) == can be used to compare the address of an object whereas, equals method is used to compare content present inside the address.
2) We can use == for the comparison of primitive data types as well as non-primitive data types, it means we can compare the values of normal variables and addresses of reference variables.

3) Equals method is mainly used for the comparison of string values not for normal values.

Q. What is the use of the equalsIgnoreCase method?

Ans:

1) This method is also used for the comparison of 2 different string values.
2) equalsIgnoreCase method compares the content of two string values irrespective of case i.e uppercase and lowercase.

Q. Why is String class immutable in nature?

Ans:

1) Immutable means non-changeable.
2) Objects are classified into 2 categories i.e mutable and immutable.
3) If an object is immutable then modifications are not possible inside the existing copy, instead of that one separate copy will be generated for the modified string object.
4) String class is immutable in nature because after re-initialisation or modification new copy will be generated
5) After generating new copy old copy will be de-referred it means address variable will not be available for old copy
6) String class is immutable; it does not mean that we cannot reinitialize the value.

Q. Explain String Functions.

Ans:

1) String functions can be used to perform various operations on String.
2) All the functions are Case-sensitive.
3) String class provides different inbuilt functions to operate string data.
    a) length(): Used to count the total number of characters(including space) in the string. It always returns integer values.
    b) charAt(int index): This function can be used to find out the character present at a specified index. This function accepts index numbers as an input and returns character output .
    c) indexOf(char ch): indexOf function can be used to find the index of a specified character. This Function always accepts character as an input and returns integer type value which represents index number. If the given character is having more than one occurrence then indexOf function will always return first occurence.
    d) lastIndexOf(char ch): This function is also used to find out the index of a specified character. If the character is having more than one occurrence then this function will return the last occurrence.
    e) contains(String charseq): contains function is used to check whether the given character sequence is present inside the string value or not. If the character sequence is present then this function will return true else it will return false.
    f) startsWith(String charseq): This function is used to check whether the given character sequence is present at start

position or not. This function also returns a boolean type of value, if the sequence is present in the starting position the output will be true else it will return false.

g) endsWith(String charseq): This function is used to check whether the given

h) character sequence is present in the ending position or not. If the sequence is present in the ending position then output will be true else output will be false.

i) subString(int startIndex): It can be used to extract a specified number of characters from the given string value. subString is also known as a given String. subString can be used with 2 types of parameters : 1. We can pass only the start index as a parameter for the subString function.
    i) Syntax: we can pass start index as well as end index as parameter,
    ii) Ex. str.subString(startIndex, endIndex);

j) toLowerCase().

k) toUpperCase().

l) toCharArray(): It converts the given String input to a character array.

m) split(): It can be used to divide the given string value into multiple parts. This function accepts a string type of parameter as a condition and always returns an array of type String.

n) isEmpty(): This function is used to check whether the given string value is empty or not. If the string variable doesn’t contain any data then this function will return output as true. If the actual value is present then output will be false.

o) trim(): trim function can be used to remove the whitespaces from leading and trailing positions. This function will always return updated string value after removing whitespace.

Q. What are the object-oriented concepts?

Ans:

1) Java is an Object-Oriented programming language.
2) By using OO Concepts we can develop different types of java applications.
3) Following are the pillars of Object-Oriented Programming.
    a) Inheritance.
    b) Polymorphism.
    c) Encapsulation
    d) Abstraction.

Q. What is meant by object composition?

Ans:

1) In java objects can have 2 types of relationships
    a) Has-A(Object Composition)
    b) Is-A(Inheritance)
2) If one object holds the properties of another object then it will be considered as object composition.
3) It is possible to composite multiple objects inside the single object.

4) Reference variables of an object will be considered as members of the class which can be static or non-static.

Q. What is an inheritance?

Ans:

1) If one class acquires the properties of another class, then it is known as an inheritance or Is-A relationship.
2) The class which acquires the properties of another class is known as subclass.
3) The class from which properties are going to inherit is known as super class.
4) Inheritance is possible by using extended keywords.
5) Sub-class can inherit non-static members of a superclass.

Q. Explain the types of inheritance.

Ans:

1) Inheritance classified into 4 types.
2) Single Level Inheritance: If one sub-class extends the properties of only one superclass then it will be considered as single level inheritance.
3) Multi Level Inheritance: If one subclass extends the properties of one superclass and that super class acquires the properties of its superclass this process will be considered multi-level Inheritance.
4) Multiple Inheritance: If one subclass is trying to extend properties of more than one superclass then it is known as multiple inheritance.

5) Hierarchical Inheritance: If one superclass is having more than one sub-classes is known as hierarchical inheritance. In other words hierarchical inheritance means multiple subclasses can extend the properties of only one superclass.

Q. What is a Constructor Call?

Ans:

1) The process of calling a superclass constructor using a subclass constructor is known as constructor call.
2) If 2 classes are having IS-A relationship then constructor call is mandatory.
3) constructor calls can be done by using super() statements.
4) super() statement should be the first statement of constructor calling.it means multiple super() statements are not allowed, because java does not support multiple inheritance.
5) Constructor calls can be done implicitly as well as explicitly.
6) Implicit constructor call:
    a) If Compiler calls the constructor by providing a super statement inside the sub-class statement in the body then it is known as implicit constructor call.
    b) A Compiler can call only zero parameterised constructor of superclass.
7) Explicit constructor call:
    a) If a programmer calls the constructor of superclass by providing a super

b) statement then it is known as an explicit constructor call.
c) If the superclass constructor is a parameterised constructor and constructor call must be done explicitly.

Q. Why does java not support multiple Inheritance?

Ans:

1) Multiple Inheritance means one subclass is acquiring more than one superclass.
2) If one class is acquiring properties of more than one superclass then we have to provide multiple super statements inside the sub class constructor body which is not possible because super statements must be the first statement of the constructor.
3) Because of all these reasons, java does not support multiple inheritance.
4) If a subclass is trying to access properties of more than one superclass then it creates diamond ambiguity.

Q. What is a Constructor Chain?

Ans:

1) Subclass constructor calls the constructor of superclass that super class constructor calls the constructor of its super class constructor, this process is known as constructor chaining.
2) It is possible only through inheritance .
3) Every class in java will have a super class either provided by developer or else compiler.

4) If the developer is not providing a super class then by default the compiler provides the object as a superclass.
5) It means every class in java will have properties of an object class.

Q. What is the use of this statement?

Ans:

1) This statement can be used to call another constructor of the same class.
2) We can use this statement in case of constructor overloading.
3) This statement should be the first statement of the constructor body.
4) Multiple statements are not allowed inside the constructor.

Q. What is the use of super keywords?

Ans:

1) Super keyword can be used to access the non-static members of superclass.
2) If the superclass and subclass have the same identifier then we can use super keyword to differentiate super class and subclass identifiers.
3) We can access multiple members by using super keywords.

Q. What is method overloading?

Ans:

1) The process of creating multiple methods with the same name but different arguments is known as method overloading.

2) At the time of method overloading, parameters must be different either by parameter length or parameter type.
3) Method Overloading can be used to achieve the same functionality, with different parameters.
4) We can overload static as well as non-static methods.
5) It is possible to achieve compile-time polymorphism, by using method overloading.
6) Subclass can overload the method of superclass if two classes are having is-A relationship.

Q. Is it possible to overload the main method?

Ans:

1) Yes, it is possible. If the class is having multiple main methods in such a case, JVM will execute the standard Main Method.
2) The method which accepts the String[] type of arguments will be considered as the Standard main method, remaining methods will be treated as external methods.
3) We have to call external main methods inside the standard main method for execution.

Q. What is method overriding?

Ans:

1) The process of inheriting a method from subclass and changing its implementation in subclass is known as method overriding.
2) In case of method overriding method signature should be the same, but method implementation can be different.

3) It is used to achieve different functionalities with the same parameters.
4) It is possible only through IS-A relationship.
5) By using method overriding we can achieve run-time polymorphism.

Q. Which methods cannot be overridden?

Ans:

1) We cannot override 3 types of methods:
2) Static: they are the class members and they will be having a single copy inside the JVM Memory hence it is not possible to override static methods.
3) Private: They are not accessible outside the class so it is not possible to override private methods.
4) Final: If the method is final, then it is not possible to change the definition of the respective method, hence we cannot override the final method.

Q. What is the use of the final keyword?

Ans:

1) Final keyword can be used to declare constant values.
2) We can use the final keyword at 3 different locations.
    a) Final variable
    b) Final Method
    c) Final Class.
3) If the variable is final then re-initialisation is not possible.

4) If the method is final then overriding is not possible.
5) If class is final then inheritance is not possible.

Q. What is Casting?

Ans:

1) The process of converting one type of information into another type is known as casting.
2) In java casting is mainly classified into 2 types:
3) Primitive casting
4) Non-Primitive casting.
5) PRIMITIVE(DATA-TYPE)
    a) NARROWING
    b) WIDENING
6) NON-PRIMITIVE(CLASS-TYPE)
    a) UPCASTING
    b) DOWNCASTING

Q. What is primitive casting?

Ans:

1) It is also known as data-type casting.
2) The process of converting one primitive information into another primitive information is known as data type casting or primitive casting.
3) Primitive casting is mainly classified into 2 types:
    a) Narrowing

    b) Widening

4) Narrowing: The process of casting higer types of information into lower types of information is known as narrowing. It must be done explicitly.
5) Widening: The process of casting lower types of information into higher types of information is known as widening. It can be done implicitly as well as explicitly.

Q. What is Class type-casting?

Ans:

1) The process of casting one class type information into another class is known as class type casting.
2) To perform class type casting we have to follow some rules:
3) 2 classes should have IS-A relationships.
4) The object which has to be casted into another class should contain the properties of the target class.
5) Class type casting is classified into 2 types:
    a) Up-casting
    b) Down-casting

Q. What is Up-casting?

Ans:

1) The process of casting subclass type information into its superclass is known as an Up-casting.
2) It can be done implicitly as well as explicitly.

3) If the compiler performs up-casting then it is known as implicit up-casting.
4) If a programmer performs up-casting the unit will be considered explicit up-casting.
5) After up-casting we can access the properties of only superclasses.

Q. What is ClassCastException?

Ans:

1) It is a type of run-time Exception or unchecked exception.
2) JVM is responsible for throwing ClassCastException.

Q. When does it occur?

Ans:

1) JVM will throw a ClassCastException if the developer is trying to convert one class type object into another class type and the respective object does not contain the properties of the target class.

Q. Why does the Compiler not throw a ClassCastException?

Ans:

1) Because the compiler is responsible for checking syntax and rules of the language.
2) Compiler will not throw a ClassCastException because syntactically it is correct.

Q. How to avoid ClassCastException?

Ans:

1) We can avoid ClassCastException by using instanceof operator.

NOTE: METHOD OVERRIDING IS THE EXCEPTIONAL CASE FOR THE RULE OF UPCASTING. It means, if a method is overridden, the latest implementation will always be executed.

Q. What is Downcasting?

Ans:

1) The process of casting superclass-type information into its subclass is known as downcasting.
2) Downcasting must be done explicitly.
3) To perform downcasting, first we have to upcast an object, it means without upcasting, downcasting is not possible.
4) After downcasting we can access the properties of subclass as well as superclass.

Q. What is Polymorphism?

Ans:

1) An object showing different behavior in different stages is known as polymorphism.
2) Polymorphism has 2 types:
3) Compile Time Polymorphism:
    a) If the compiler binds the method declaration along with its definition at the time of compilation is known as compile-time polymorphism.

b) Compile Time Polymorphism is also known as early binding or static binding.
c) Static binding cannot be re-binded.
d) Method Overloading can be used to achieve compile time Polymorphism.

4) Run-time Polymorphism:
   a) If JVM binds the method declaration and definition at the time of execution then it is known as run-time polymorphism.
   b) Run-time Polymorphism is also known as late binding or dynamic binding.
   c) Dynamic binding can be rebonded.
   d) To achieve Run-time Polymorphism we have to use following concepts:
      i) Method Overriding.
      ii) Inheritance
      iii) Up-casting.
   e) By using Run-time Polymorphism we can achieve generalization.

Q. What is Generalization?

Ans:

1) It is the process of extracting common properties of all classes and combining them into one class.
2) Generalization can be achieved through run-time polymorphism.

Q. What is an abstract class?

Ans:

1) The method which has a declaration as well as definition is known as a concrete method.
2) The class which contains concrete methods is known as concrete class.
3) If a method is having only a declaration then it will be considered as an abstract method.
4) If a method is an abstract then it must be declared inside the abstract class.
5) Abstract class can act as a container for abstract as well as concrete methods.
6) Abstract methods and classes must be declared by using abstract keywords.
7) It is not possible to create an object of an abstract class, because a new operator does not work with an abstract keyword.
8) A subclass can extend the properties of a super class which is abstract in nature, in such a case, the sub-class is responsible to provide an implementation for all the abstract methods of the superclass. If not then the sub-class will also become an abstract class.
9) Abstract class can be used to achieve partial abstraction.

Q. What is an Interface?

Ans:

1) Interface is a java type definition block which can be used to declare abstract methods.
2) We can not provide concrete methods inside the interface because it is completely abstract in nature.
3) It is possible to declare static concrete methods from JDK1.8.
4) It is possible to declare data members inside the class and by-default all the data members will be having public static and final access.
5) A class can implement properties of interface.
6) In such a case class is responsible to provide implementation for all the abstract methods of interface. If not then class will become an abstract class.
7) It is not possible to create an object of interface, because it contains only abstract methods.
8) One interface can extend the properties of another interface.
9) We can create a reference variable of type interface in such a case, we have to provide an object of implementation class.
10) One class can implement the properties of more than one interface.

Q. Why is it not possible to create an object of Interface?

Ans:

1) Interface is a java type definition block which does not extend the properties of Object Class.
2) We can not create an object of Interface, because it contains no-static abstract methods.

3) At the time of object creation we have to use a new operator which does not work with abstract keywords.
4) Since Interface is a definition block, which does not contain constructor and object creation is not possible without constructor.
5) Because of all these reasons we cannot create objects on the Interface.

Q. How to achieve multiple Inheritance by using Interface?

Ans:

1) The class can implement properties of more than one interface because the interface does not extend the properties of the object class.
2) To achieve multiple inheritance we have to create multiple interfaces and all interfaces should be implemented by using java class.
3) We can extend a class and implement an interface by using a single sub-class.

Q. What is meant by access modifier in Java?

Ans:

1) Access Modifiers can be used to set the visibility for the members of the class.
2) In java access modifiers are mainly classified into 4 types:
3) Private: If members are having private access then we cannot access those members outside the class.

4) Package-Level: If members are having package level access then we can access those members anywhere inside the same package.
5) Protected: protected members can be accessible anywhere inside the same package as well as outside the package only through Inheritance.
6) Public: we can access public members anywhere inside the project as well as outside the project.

Q. What is Encapsulation?

Ans:

1) The process of binding members of the class along with a class body and protecting them by using access modifiers is known as an encapsulation.
2) In other words encapsulation means the process of wrapping data members and function members as a single unit.
3) Encapsulation can be used to achieve data hiding.

Q. Rules of Encapsulation.

Ans:

1) Section-1:
    a) We Can Declare One class inside the other class, then it will be considered as an inner class.

   b) Inner class can have any of 4 access modifiers whereas outer class can have either public or package level access.
   c) By default all the members of class will be having package level access whereas interface members will be having public level access.
2) Section-2:
   a) One java file can act as a container for multiple classes. In such a case only one class should have public level access. File name should be same as class name which is having public access
   b) If all classes are having package level access then we can provide any class name as a filename.
3) Section-3:
   a) A constructor can have anyone of 4 access modifiers.
   b) Default constructor will be having the same access as that of a class.
   c) If the constructor is private then object creation is not possible.

Q. What is a Java Bean Class?

Ans:

1) The class which contains private data members and public getter, setter.

2) Java Bean class can be used to achieve encapsulation along with data hiding.
3) Getters method in the class provides the read access for private data members. It means we can access private members outside the class through the getter method.
4) Setter method provides write access for private data members, it means we can modify private data members outside the class by using the setters method.

Q. What is an inner class?

Ans:

1) One class declared inside the other class known as inner class or nested-class.
2) Inner class can be used to encapsulate properties of one object inside the other object.
3) It is possible to create multiple inner classes inside the single outer class.
4) Inner class can be static or non-static because the inner class will be treated as a normal member of the class.
5) To access properties of the inner class we have to create an object of the outer class.
6) Syntax:

```
class Outer
{
        static class inner
```

```
          {
          -----;
          }
}
class outer
{
          class inner
          {
          ----;
          }
}
```

Q. What is java anonymous class?

Ans:

1) In java we can declare one class inside the other class, then it will be considered an Inner class.
2) The inner class declared without any name is known as anonymous class.
3) We can create anonymous classes that provide implementation to all abstract methods of an interface at the time of object creation.
4) After creating an anonymous class we don't have to provide implementation for abstract methods of an interface by using sub-classes.

Q. What is a Singleton Design Pattern?

Ans:

1) The process of creating only one object throughout the application of life cycle is known as the singleton design pattern.
2) The class which contains a single tone object is known as a singleton class.
3) To create a singleton object we have to declare a private constructor and one static method for object creation.
4) If an object is created then we have to provide the address of that object to the new reference variable and if the object is not present then the new object will be created.

Q. What is an Abstraction?

Ans:

1) Abstraction is the process of hiding an implementation of an object without exposing it to the utilisation layer.
2) In other words abstraction means hiding object functionality and providing an interface or media to access that functionality.
3) Abstraction can be achieved by using 3 steps:
    a) Generalise all the properties of an object into an interface.
    b) Provide implementation for object functionalities by using classes.
    c) Create a reference variable of type interface to access object functionalities.
4) Abstraction can be used to achieve loose-coupling.

Q. What is coupling?

Ans:

1) Coupling represents dependency between 2 classes.
2) In java there are 2 types of coupling.
    a) Loose Coupling
    b) Tight Coupling
3) If one class is independent of another class then it is known as loose-coupling.
4) If one class is completely dependent on another class then it is known as tight coupling.
5) Example:
    a) super class→ Car(TIGHT COUPLING)
    b) Sub class→ vehicle.
    c) Super class→ Vehicle(LOOSE COUPLING)
    d) Subclass→ Bike,Car,etc

Q. What is the Factory-Design Pattern?

Ans:

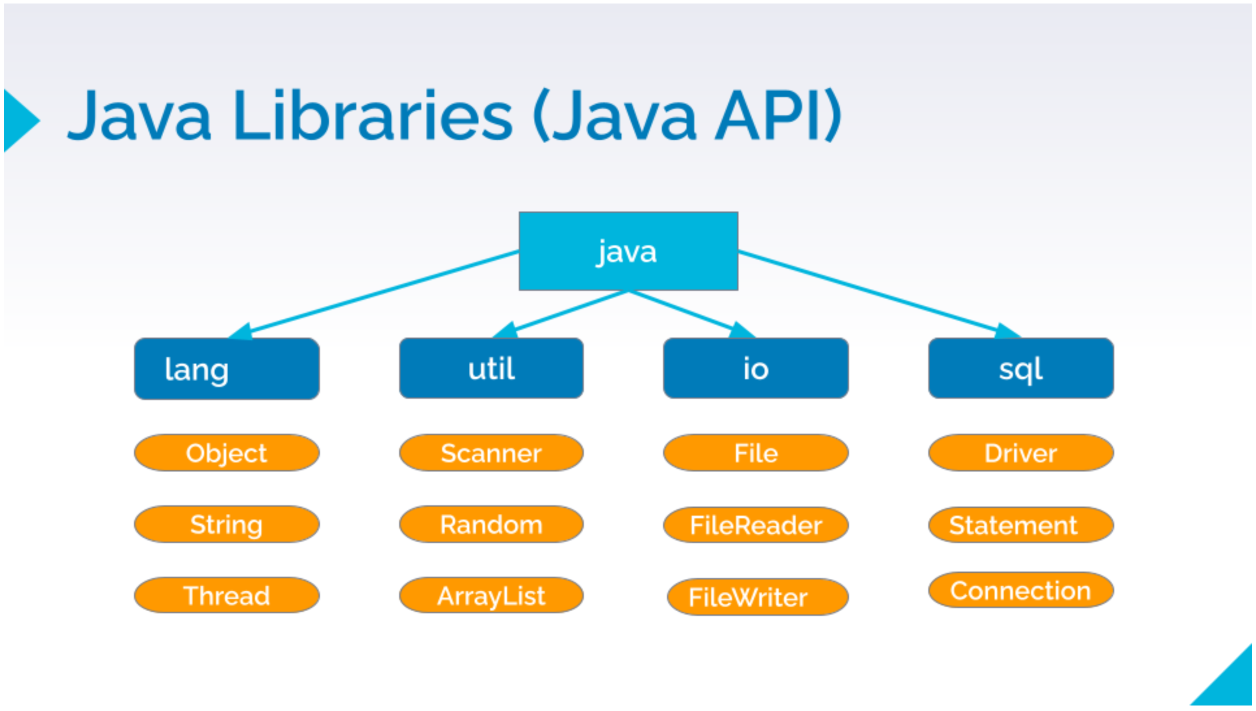
1) The process of creating objects of implementation classes without exposing it to the utilisation layer is known as factory design pattern.
2) The method which contains objects of implementation classes is known as the factory method.

3) The class which contains the factory method is known as factory class.
4) Abstraction is the backbone of factory-design patterns.
5) By using factory design patterns we can create Loosely Coupled Objects.

Q. Explain Java Libraries.

Ans:

1) Java Library is a collection of Multiple Classes and interfaces.
2) It is also known as API, because it provides the platform for application
3) development.
4) Java Language has provided various packages which can be used to develop different types of java applications.
5) Eg: DataBase Application, File Applications, Collection Applications, etc.
6) Language package is the default package for every java application. It means every java program will have properties of the language package.
7) To use the features of other packages, we have to import them explicitly.



Java Libraries (Java API)

Q. Explain Object Class.

Ans:

1) It is declared inside the java.language package.
2) Every java class will have properties of an object class.
3) A developer can inherit properties of an object class implicitly as well as explicitly.
4) Object class has provided few important methods which can be used to get the different types of information.
    a) toString()
    b) hashCode()
    c) clone()
    d) equals(String args)
    e) getClass()

f) finalize()

g) wait()

h) notify()

i) notifyAll()

Q. Explain the use of the toString method.

Ans:

1) toString method is declared inside the object class.
2) This method can be used to represent an object as a string value.
3) For the string representation of an object we have to override the toString method inside the respective class.
4) toString method always returns a String type of value.
5) Whenever we are printing a reference variable internally the toString method will be executed and by default it will return the address of an object.

Q. What is the hashCode method?

Ans:

hashCode method is declared inside the object class.

hashCode method can be used to identify a specific object.

Every object will be having a specific hashcode value which is present in integer format.

hashCode value will be generated by JVM immediately after object creation.

Q. What is the equals method?

Ans:

1) Equals method is also declared inside the object class.
2) This method can be used for the comparison of 2 objects.
3) At the time of comparison the hashcode of an object plays an important role.
4) If 2 objects are having the same hashcode then output will be true else output will be false.

Q. What is the getClass method?

Ans:

1) It is declared inside the object class.
2) This method always returns a full-qualified class_name of an object.
3) We can call this method by using the reference of a specific object.

Q. Explain Wrapper Classes.

Ans:

1) Every primitive data type will have a wrapper class.
2) All wrapper classes extend the properties of object class.
3) Wrapper classes are declared inside the java.language package.
4) Wrapper classes are also known as non-primitive data types which can be used to represent information in terms of objects.

Q. What is Autoboxing?

Ans:

1) The process of converting primitive information into non-primitive(Object Form) is known as autoboxing.
2) Autoboxing is applicable only for wrapper classes.
3) After autoboxing we can use the features of object class.
4) At the time of autoboxing we have to create the reference variable of a specific wrapper class.
    a) Primitive Info:
        i) int a = 20;
    b) Non-primitive Info:
        i) Integer i = new Integer(20);
    c) Autoboxing:
        i) int a =50;
        ii) Integer i = a OR Integer i = new Integer(a);

Q. What is Auto-Unboxing?

Ans:

1) The process of converting non-primitive information to primitive is known as auto unboxing.
2) Unboxing can be done only in wrapper class.
3) After unboxing we cannot use the features of object class.
4) Example:
    a) Integer i = new Integer(50);
    b) int a=i;

Q. Explain Parsing Techniques.

Ans:

1) It is used for string value conversion.
2) We can parse the string values by using wrapper class.
3) It is possible to convert string value into the specific primitive and non-primitive data type.
4) Syntax:
    a) String to primitive or non-primitive:
        i) Wrapper_class.parse()
    b) Similarly we can convert primitive data type value to the string value by using toString() method.
5) Following are the methods that can be used to convert string values to the primitive data type.
    a) Integer.parseInt(String s)
    b) Double.parseDouble(String s)
    c) Float.parseFloat(String s).
    d) Long.parseLong(String s).
    e) Short.parseShort(String s)
    f) Boolean.parseBoolean(String s)
    g) Byte.parseByte(String s)
6) To convert primitive data values in string values we can use following methods:
    a) Integer.toString(int a).
    b) Double.toString(double b).
    c) Float.toString(float c).
    d) Long.toString(long a).
    e) Short.toString(short a).

f) Boolean.toString(boolean b).

g) Byte.toString(byte b).

h) Character.toString(char c).

Q. Why do we have to use exception handling?

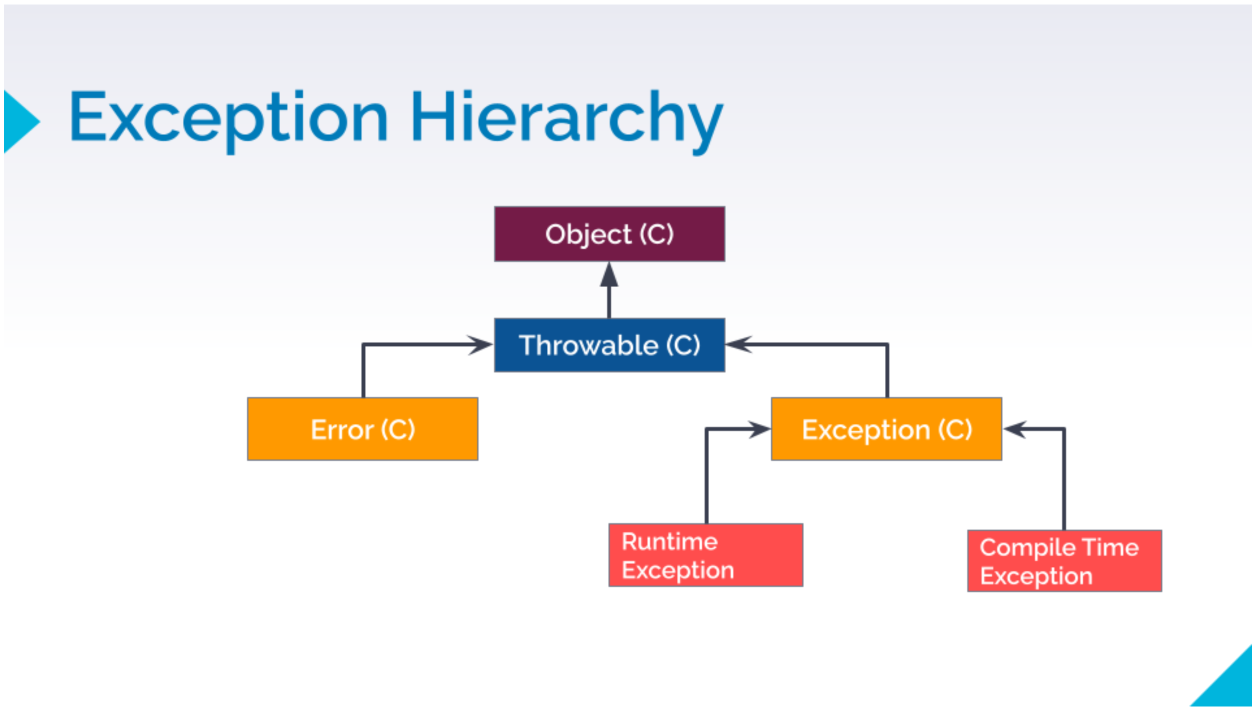
Ans:

1) Exception handling can be used to handle abnormal situations present inside the java program.
2) If we are not using exception handling then the java application will be terminated abnormally.

Q. Explain the Exception Hierarchy in java programming language.

Ans:

1) Exception hierarchy is a collection of different classes.
2) Throwable is the root class, present inside the exception hierarchy which extends the properties of an object class.
3) Throwable class is having 2 important sub-classes:
    a) Error
    b) Exception
4) Exception further classified into 2 types:
    a) Run-time Exception(Unchecked Exception).
    b) Compile-time Exception(Checked Exception).



Exception Hierarchy

Q. What is an Error?

Ans:

1) Error indicates a serious problem due to lack of system resources.
2) Error is a subclass of throwable class.
3) Examples:
    a) StackOverflow error.
    b) OutOfMemory error.

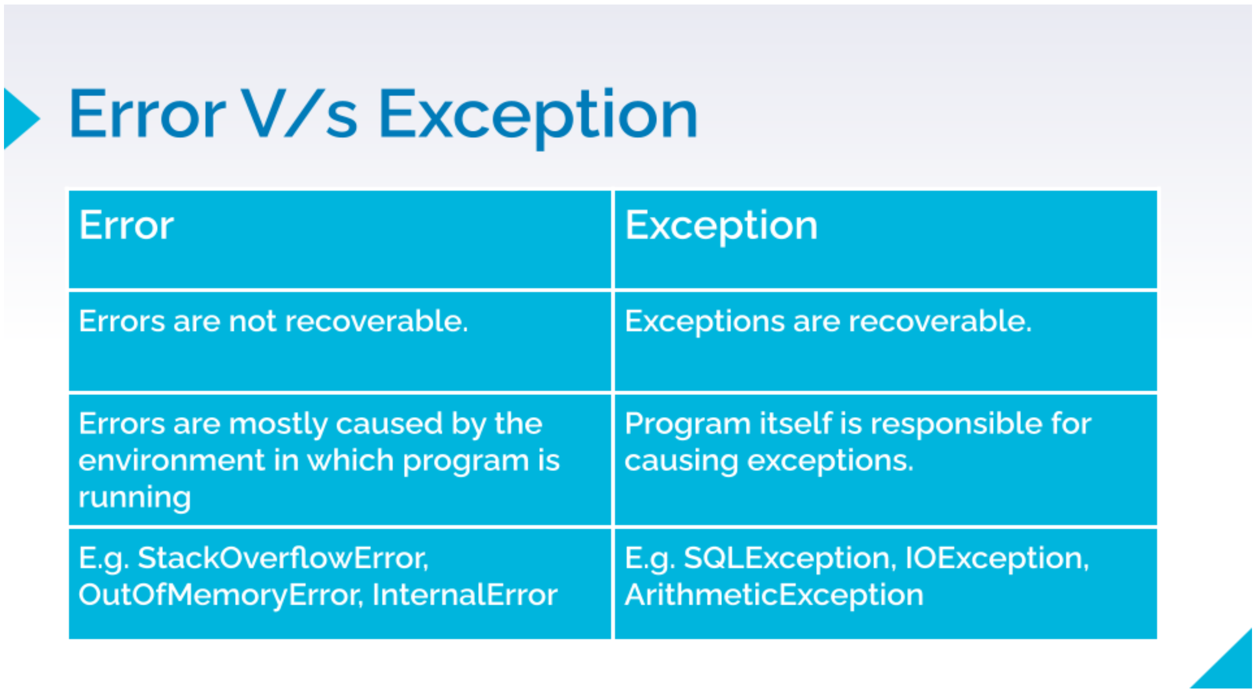
Q. What is an Exception?

Ans:

It is an abnormal situation or unwanted event which occurs during the execution of program and disrupts normal flow of the program instruction

Q. What is the difference between Error and Exception?

Ans:

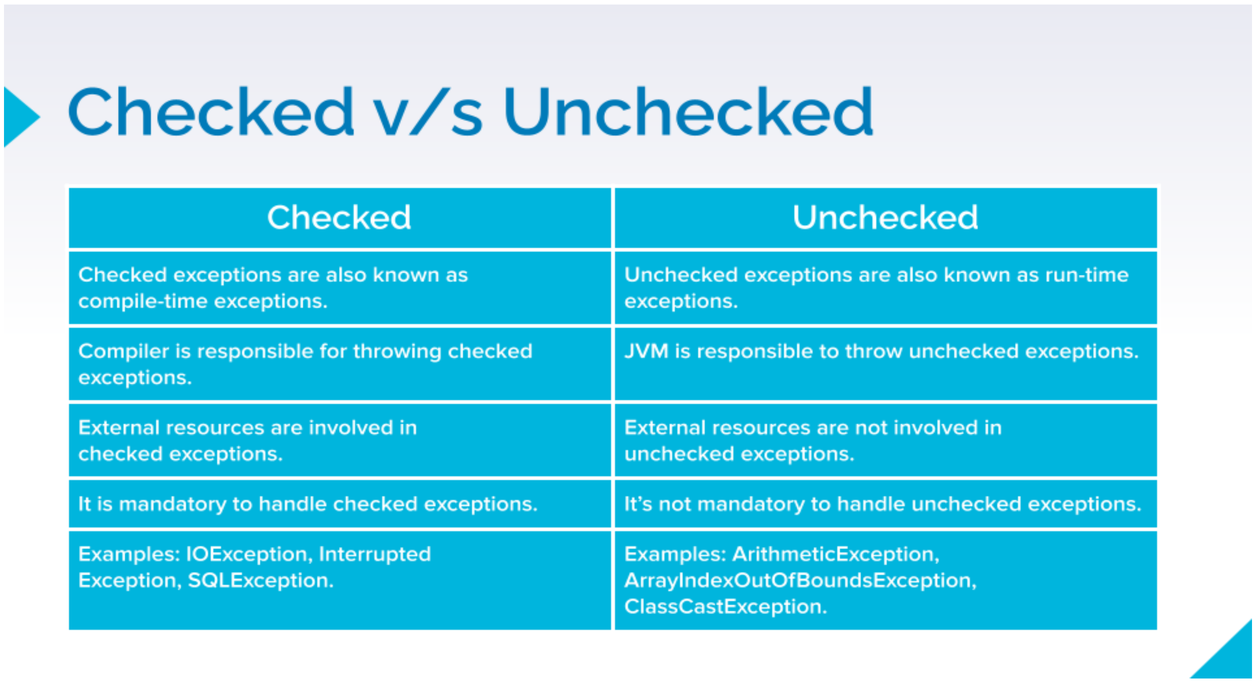


Error V/s Exception

Error	Exception
Errors are not recoverable.	Exceptions are recoverable.
Errors are mostly caused by the environment in which program is running	Program itself is responsible for causing exceptions.
E.g. StackOverflowError, OutOfMemoryError, InternalError	E.g. SQLException, IOException, ArithmeticException

Q. What is the difference between Checked and Unchecked Exceptions?

Ans:



Checked v/s Unchecked

Checked	Unchecked
Checked exceptions are also known as compile-time exceptions.	Unchecked exceptions are also known as run-time exceptions.
Compiler is responsible for throwing checked exceptions.	JVM is responsible to throw unchecked exceptions.
External resources are involved in checked exceptions.	External resources are not involved in unchecked exceptions.
It is mandatory to handle checked exceptions.	It's not mandatory to handle unchecked exceptions.
Examples: IOException, Interrupted Exception, SQLException.	Examples: ArithmeticException, ArrayIndexOutOfBoundsException, ClassCastException.

Q. What is Exception Handling?

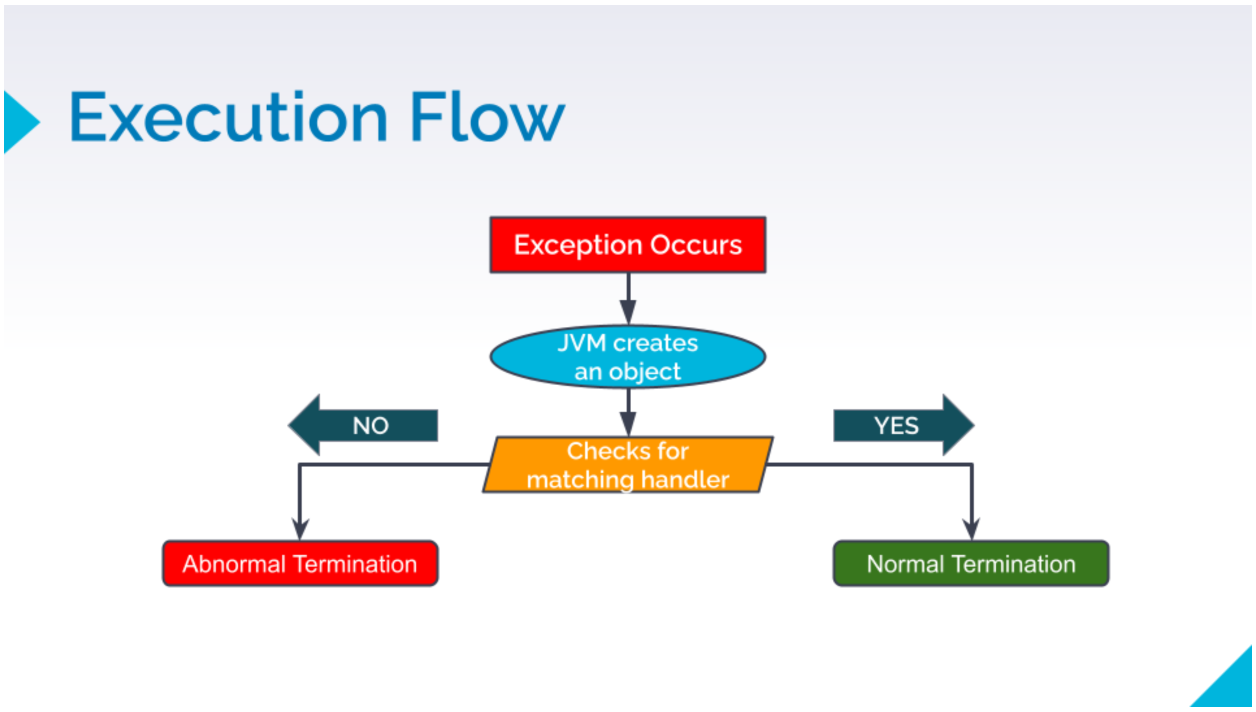
Ans:

1) Exception handling is the mechanism which is used to handle different types of exceptions.
2) This mechanism provides different blocks which can be used to recover from exceptions.
3) Exception handling can be managed by using five components
    a) try, catch, throw, throws, finally.

Q. Explain execution flow of Exception Handling.

Ans:

1) Whenever an exception occurs in a java program JVM will create an object of the respective exception class.
2) Try block is responsible for throwing an exception object created by JVM towards the catch block.
3) Inside the catch block we have to provide a matching handler to handle exception objects thrown by try blocks.
4) If a matching handler is present then the exception object will be handled and normal termination will take place.
5) If a matching handler is not provided then the program will be abnormally terminated.



Q. How to use multiple catch statements with a single try block?

Ans:

1) It is possible to provide multiple catch blocks with different handlers along with a single try block.
2) Inside the try block we can provide multiple java statements, which may cause an exception.
3) If the developer does not have an exact idea about the exception object then we can declare multiple catch blocks.
4) In case a try block is having multiple catch blocks then at a time only one catch block will be executed because one try block can throw only one exception.
5) If the try block is having multiple catch blocks then the subclass catch block must be declared before the superclass catch block.

Q. What is the use of a nested try-catch block?

Ans:

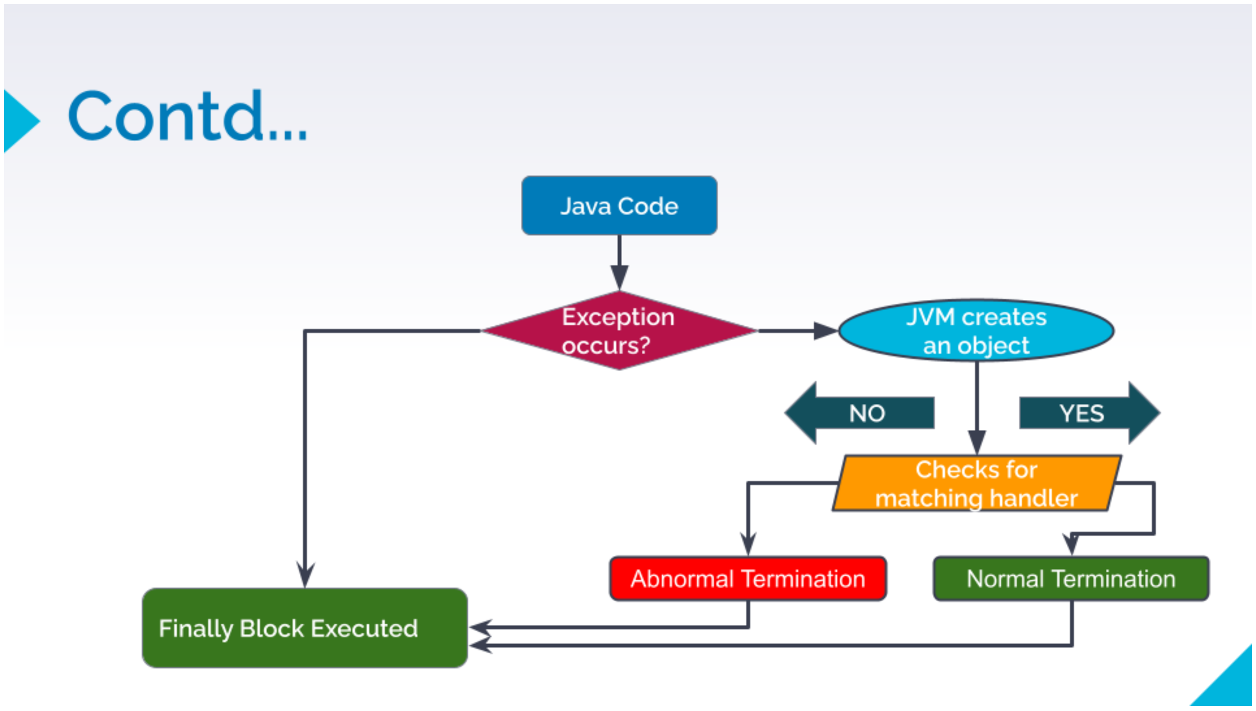
1) One try block will be able to throw only one exception object, it means we can handle objects of only one exception by using catch blocks.
2) To handle multiple exception objects, we can use nested try-catch blocks.
3) If the try-catch block is declared inside the other try block then it is known as a nested try block.
4) It is possible to declare multiple nested try-catch blocks inside the outer try block.
5) It is recommended to start the nested try block from the very first line of the outer try block.

Q. What is the use of Finally block?

Ans:

1) Finally is the special block available in exception handling Mechanism.
2) Finally block can be used to close all the costly resources.
3) Finally block will be executed irrespective of exception scenarios.
4) It means this block will be executed in every possible situation so it does not matter whether an exception occurs or not.

5) All system resources will be considered as costly resources, which can be used for inter-application communication.
6) Finally block must be declared after try or else catch block.
7) We cannot provide multiple finally blocks after try-catch blocks.
8) The Finally block would not be executed only if System.exit() is invoked before it.



Q. What is Exception propagation?

Ans:

1) If an object is moving from one point to another point then it is known as
2) propagation.
3) In case of exception handling exceptions thrown from top of the

stack will drop down to the call stack, it means an exception thrown from one method will be forwarded to another method.

4) This process will be continued until the exception gets handled or else it reaches the main method.
5) The process in which an exception object moves from one method to another method is known as exception propagation.

Q. What is the use of printStackTrace()?

Ans:

1) printStackTrace is the non-static method declared inside the throwable class, which can be used for the diagnosis of exceptions.
2) This method will print detailed information of the exception object. For example:
   a) exception name, exception class line number,etc.
3) We can call this method inside the catch block by using the reference of the exception object.

Q. What is the use of the throw keyword?

Ans:

1) Throw keyword is part of the exception handling mechanism.
2) Throw keywords are mainly used to throw exceptions.
3) Developers can throw custom exceptions with the help of throw keywords.
4) It is mandatory to provide an exception class object along with a

throw keyword.

Q. What is a Custom Exception?

Ans:

1) Custom Exception is also known as user-defined exception.
2) Apart from Java Exception classes, developers can create their own exception classes.
3) If a developer is creating an exception class, then it is known as custom-exception.
4) Developer can create 2 types of exception:
    a) Checked Exception.
    b) Un-checked Exception.
5) To create checked custom exceptions we have to extend properties of Exception class.
6) To create custom unchecked exceptions we have to extend properties of the RuntimeException class.

Q. What is the use of the throws keyword?

Ans:

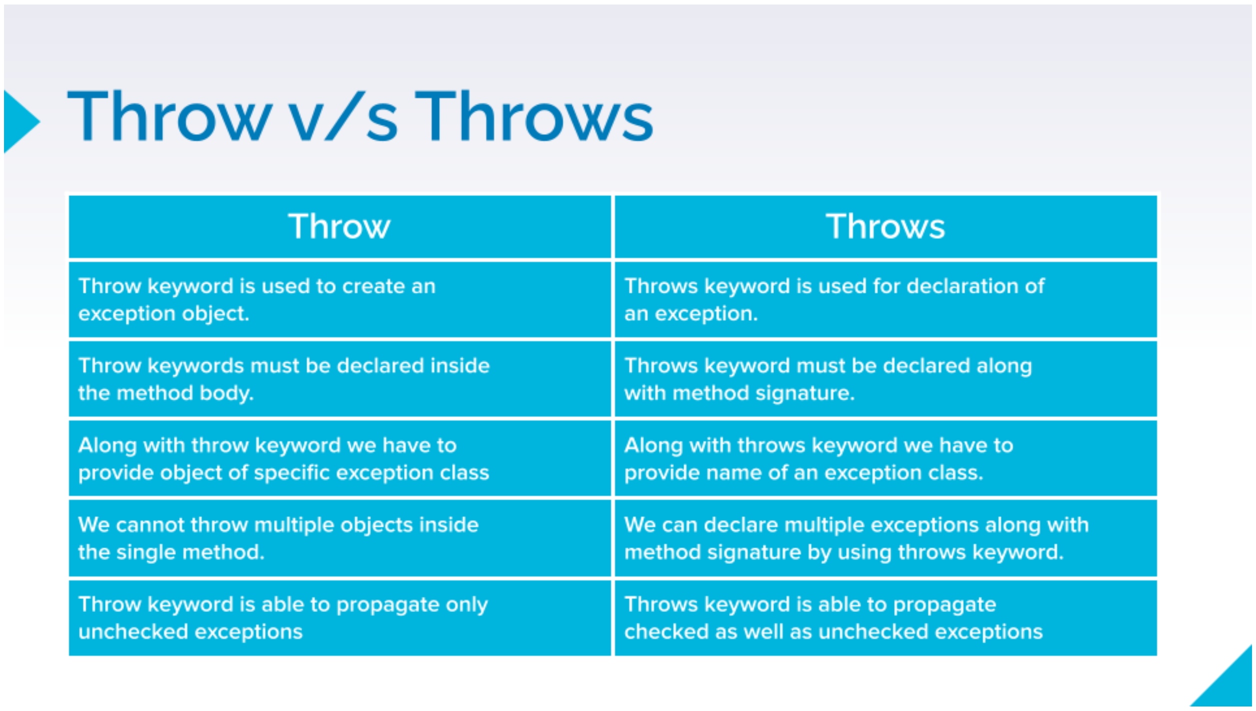
1) Throws keyword is a part of the Exception handling mechanism.
2) Throws keyword can be used to declare an exception of a specific class.
3) Exception Declaration means we have to provide an exception class

name along with a throws keyword.

4) Throws keyword plays an important role, in case of propagation of checked exceptions.
5) If you are using throws keyword then no need to provide try-catch block inside that particular method.

Q. Explain the difference between Throw and Throws keywords.

Ans:



Throw v/s Throws

Throw	Throws
Throw keyword is used to create an exception object.	Throws keyword is used for declaration of an exception.
Throw keywords must be declared inside the method body.	Throws keyword must be declared along with method signature.
Along with throw keyword we have to provide object of specific exception class	Along with throws keyword we have to provide name of an exception class.
We cannot throw multiple objects inside the single method.	We can declare multiple exceptions along with method signature by using throws keyword.
Throw keyword is able to propagate only unchecked exceptions	Throws keyword is able to propagate checked as well as unchecked exceptions

Q. What is a Custom Exception?

Ans:

1) In the Java language developers can create their own exceptions which are known as custom exceptions.

2) Developers can create run-time as well as compile-time exceptions.
3) If inbuilt exception classes are not able to handle exceptions then we have to create custom exception classes.

Q. What is the difference in collection, collections, and collection frameworks?

Ans:

1) Collection is an interface available in the java.util package.
2) Collections is a utility class which contains different methods.
3) Collection framework is a collection of interfaces and classes which can be used to manipulate data.

Q. What are the drawbacks of arrays?

Ans:

1) Arrays can be used to store only the same type of data i.e homogeneous data.
2) Array variables are fixed width or static width, which means the array is not resizeable.
3) Since arrays are not resizeable there is a chance of memory wastage.
4) Array manipulation is a time consuming process because the array does not have an add or remove method.

Q. What is a Collection?

Ans:

1) Collection is a group of multiple objects.
2) In other words an entity which acts as a container for multiple objects, is known as a collection.
3) It is possible to store the same as well as different types of information inside the collection object, it means collection classes and interfaces are capable of storing homogeneous and heterogeneous information.
4) We can efficiently process and store the data by using a collection framework.

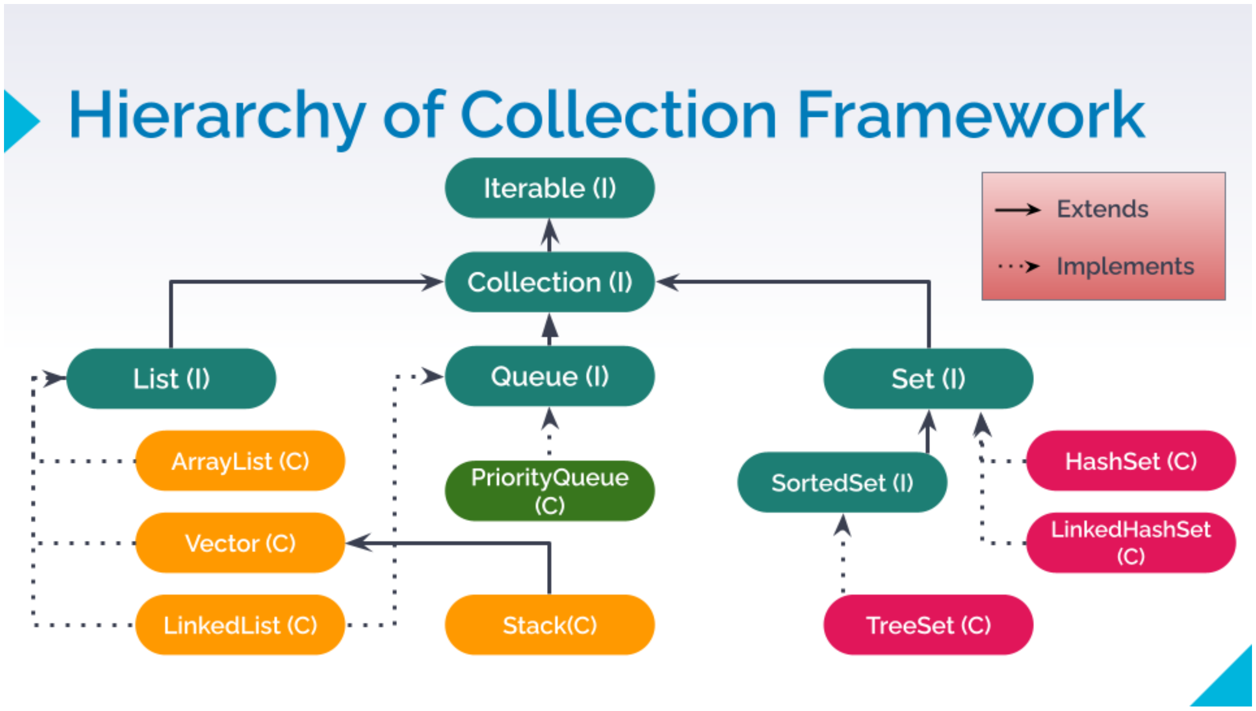
Q. What are the Benefits of the collection framework?

Ans:

1) Collection framework is resizable in nature; it means dynamic structure.
2) It increases reusability and interoperability.
3) Reduced development effort by using core collection classes rather than implementing our own classes.
4) Collection framework will reduce efforts for code maintenance.

Q. Explain Collection Framework Hierarchy.

Ans:



Q. Explain List interface in detail.

Ans:

1) List interface is a type of collection interface.
2) List interface is declared in the java.util package.
3) List interface can be used to store the data in sequential order.
4) List interface allows to store null and duplicate values.
5) Following are the implementation classes of list interface:
    a) ArrayList
    b) Vector
    c) LinkedList

Q. Explain ArrayList class.

Ans:

1) ArrayList is an implementation class for list interface.
2) The ArrayList class is declared inside the java.util package.
3) ArrayList internally uses the concept of dynamic array to store the elements.
4) Since it is an implementation class of List interface data will be stored in a sequential manner.
5) ArrayList allows storing null as well as duplicate values.
6) ArrayList implements following marker interfaces:
    a) Serializable.
    b) Clonnable.
    c) RandomAccess.
7) Default capacity of ArrayList is 10 elements.
8) In ArrayList capacity will grow and shrink dynamically which means if you are going to insert elements in ArrayList memory will be allocated automatically, similarly memory will be released if you are going to remove elements from ArrayList.
9) Formula:
    a) new capacity = (old capacity * 3)/2+1 (till JDK 1.6)
    b) new capacity = (old capacity * 3)/2(from JDK 1.7)

Q. Enlist the methods of the ArrayList class.

Ans:

1) add(Object o):
    a) This method can be used to add a new element inside the array List.
    b) Every element should be represented in the form of an object.
2) add(int index, Object o):
    a) This method can be used to add a new element at a specific position.
3) remove(Object o):
    a) This method can be used to remove specific elements from the arraylist.
4) remove(int index):
    a) This method always removes the elements from the specified position.
5) set(int index, Object o):
    a) This method is used to modify the element present at specified index.
6) indexOf(Object o):
    a) This function can be used to get the index number of the specified element.
7) get(int Index):
    a) This function always returns an element present at a specified

index.

8) size():
    a) This function always returns the capacity of arrayList.
9) contains(Object o):
    a) This function always checks whether a specified element is present or not inside the arrayList.

Q. What is the difference between generic and non generic arrayList?

Ans:

1) Generic array list allows to store only specific types of information or elements. Whereas non generic allows to store the same type as well as different types of elements.
2) To create a generic arraylist we have to specify the respective class name, whereas to create a non-generic array list we don't have to specify any class name.

Q. What is the difference between iterator and ListIterator?

Ans:

1) Both are the interfaces declared in the java.lang package, which can be used to iterate collections.
2) The Iterator interface can be used for traversing lists as well as set whereas ListIterator interface is used for traversing lists.
3) The Iterator interface allows traversing the collection only in a

forward direction, whereas ListIterator supports forward as well as backward traversing.

4) We can not add a new element to a collection while traversing using an iterator whereas ListIterator allows us to add a new element while traversing.

Q. Explain Vector class.

Ans:

1) Vector is a legacy class declared in the java.util package.
2) It is an implementation class of List interface so it allows us to store data in sequential manner.
3) It is possible to store null and duplicate values inside the vector.
4) Vector is similar to ArrayList only the difference is that ArrayList is thread unsafe whereas Vector is thread safe.
5) It means ArrayList is asynchronous but vector is synchronous.
6) Vector gives poor performance in insert, update, delete or select operations as compared to ArrayList.
7) Vector has a default capacity of 10 elements.
8) Formula:
    a) new capacity = old capacity * 2.

Q. Explain the working of Stack class.

Ans:

1) Collection framework has provided stack class to implement features of stack data structure.
2) Stack class is declared in Java.util package
3) Stack class extends the properties of vector class.
4) Stack class stores the data in the First In Last Out manner.
5) Methods of stack
    a) push () : this method will insert a new element at the top of the stack.
    b) pop(): this method will remove elements from top of the stack.
    c) peek(): this method always returns an element from top of the stack.
    d) search (): this method can be used to search an element present inside the stack. It returns the position from top of the stack.
    e) empty(): this method will check whether the stack is empty or not.

Q. Explain LinkedList class.

Ans:

1) LinkedList is a linear data structure.
2) It implements the properties of List as well as Queue interface.
3) LinkedList elements will not be stored in the array however they are linked with each other using pointers.

4) LinkedList internally uses a Doubly Linked List data structure.

5) Each element in the Linked List is known as a node.

6) It doesn't have a default capacity.

Q. Enlist the methods of LinkedList.

Ans:

1) addFirst(Object o)

2) addLast(Object o)

3) removeFirst()

4) removeLast()

Q. What is the difference between LinkedList and ArrayList?

Ans:

ArrayList	LinkedList
It implements properties of List interface.	It implements properties of List as well as Queue Interface.
ArrayList Internally uses the concept of dynamic array.	LinkedList internally uses the concept of doubly linked list.
Data will be stored in terms of indexes.	Data will be stored in terms of nodes.

ArrayList has a default capacity of 10 elements.	LinkedList does not have default capacity.
ArrayList gives the best performance for searching and sorting operations.	LinkedList gives the best performance for insert, update and delete operations.

Q. Explain the features of Set interface.

Ans:

1) Set interface is one of the important types of collection interface.
2) Set interface can be used to create a group of unique objects.
3) It means we can not store duplicate values inside the set.
4) Set interface doesn't maintain the order of insertion because it stores the data in random order.
5) Following are the types of Set implementations
    a) HashSet
    b) LinkedHashSet
    c) TreeSet

Q. Explain HashSet class.

Ans:

1) HashSet is an implementation class of Set interface.
2) It doesn't maintain order of insertion means data will be stored in

random order.

3) HashSet doesn't allow you to store duplicate values.
4) HashSet internally uses HashTable data structure to store objects.
5) HashSet has a default capacity of 16 elements with load factor 0.75.
    a) Threshold = old capacity * 0.75
    b) New capacity = old capacity *2
6) Once the HashSet cross threshold limit its capacity becomes double.

Q. Explain LinkedHashSet class.

Ans:

1) LinkedHashSet is an implementation class of Set interface.
2) It is the combination of list features and set features.
3) This class allows storing the data in sequential order but doesn't allow duplicate values.
4) LinkedHashSet also has a default capacity of 16 elements with load factor 0.75.

Q. Explain TreeSet class.

Ans:

1) The TreeSet class is similar to the HashSet class.
2) TreeSet implements SortedSet interface which is the part of Set interface.

3) This class is mainly used for sorting set elements.
4) TreeSet always sorts the data in ascending order.
5) TreeSet allows storing only generic elements.
6) TreeSet also has a default capacity of 16 elements with load factor 0.75.

Q. Explain Queue Interface.

Ans:

1) Queue interface works on FIFO principle.
2) This interface is designed in such a way that new elements inserted will be added at the end of the queue and removed from the beginning.
3) Queue interface has two implementation classes.
   a) LinkedList.
   b) PriorityQueue.

Q. Enlist the important methods of Queue interface.

Ans:

1) element(): This method returns the head of the queue(first element).

2) peek(): This method also returns the head of the queue. If the queue is empty then it returns null.
3) remove(): This method is used to remove the head of the queue.
4) poll(): This method also removes the head of the queue. If the queue is empty then it returns null.
5) offer(Object o): This method is used to add a new element inside the queue.

Q. Explain PriorityQueue class.

Ans:

1) PriorityQueue is an implementation class of Queue interface.
2) Unlike normal queues PriorityQueue elements are retrieved in sorted manner.
3) It means the head of the queue is the smallest element.

Q. Explain Map Interface in detail.

Ans:

1) Map interface is mainly used to store the data in key-value pairs.
2) Map interface does not extend the properties of collection interface

but still it is a part of collection framework.

3) A map can not have duplicate keys but duplicate values are allowed.
4) Each key should be associated with a maximum one value.
5) Each key value pair in the map will be stored as a Map.Entry object.
6) Map interface internally uses HashTable data structure to store objects.
7) Default capacity of Map is 16 elements with load factor 0.75.
8) Following are the types of Map
    a) HashMap
    b) LinkedHashMap
    c) TreeMap

Q. Enlist the important methods of Map interface.

Ans:

1) size()
    a) Returns the total number of entries present inside the Map.
2) isEmpty()
    a) It checks whether the Map is empty or not.

3) containsKey(Object key)
    a) It checks whether the Map contains a specified key or not.
4) containsValue(Object value)
    a) It checks whether the Map contains a specified value or not
5) get(Object key)
    a) Returns the value associated with the specified key.
6) put(Object key, Object value)
    a) Adds a new key value pair inside the Map.
7) remove(Object key)
    a) It removes the specified key along with its value.
8) putAll(Map o)
    a) It copies all entries from one Map into another Map.
9) clear()
    a) It removes all elements from the Map.
10) keySet()
    a) Returns the set containing all the keys of Map.
11) values()
    a) Returns the collection of all the values present inside the Map.
12) entrySet()

a) Returns the set of all the entries from the specified Map.

Q. Explain the working of HashMap.

Ans:

1) It is an implementation class of Map interface.
2) Since it implements Map, All the properties of Map interface are present inside the HashMap.
3) HashMap stores the data in random order.
4) Default capacity of HashMap is 16 elements with load factor 0.75.

Q. Explain the working of LinkedHashMap.

Ans:

1) It is an implementation class of Map interface.
2) Since it implements Map, All the properties of Map interface are present inside the LinkedHashMap.
3) LinkedHashMap stores the data in sequential order.
4) Default capacity of LinkedHashMap is 16 elements with load factor

0.75.

Q.Explain the working of TreeMap.

Ans:

It is an implementation class of SortedMap interface.

Since it implements Map, All the properties of Map interface are present inside the TreeMap.

TreeMap stores the data in a sorted manner.

Default capacity of TreeMap is 16 elements with load factor 0.75.