**Quest 1:**

**Object-oriented programming (OOP):**

Object oriented programming is a programming model where program is compartmentalized based on objects. Program is divided into smaller modules based on data and functions. This helps programmers to connect with real time scenario and helps making program easier. This concept helps in reusability and simplification of code.

**Examples of OOP:**

* + Abstraction
  + Encapsulation
  + Inheritance
  + Polymorphism

**Quest 2:**

**Access Specifiers:**

Access specifiers are keyword used in object oriented programming for the purpose of encapsulation. It regulates access to classes, fields, and methods whether to be used or invoked from another method, class or package.

**Types of Access Specifiers:**

* + Public
  + Private
  + Protected
  + Default(no specifier)

**Quest3:**

**Static and its requirement:**

Static keyword is mainly used for memory management. Static variable gets memory only once in class area at the time of class loading. It saves memory. Static method or variable does not require any instance creation (object creation). That’s why main method is a static method and we don’t create an object for main method and program starts from main method.

**Quest4:**

**Array:**

It is a collection of related elements in an ordered arrangement. It groups data of same data-type into a single object. It store multiple data of same data-type in a single variable.

**Quest5:**

**Data Type:**

Data type is the classification of variables based on the type of data that the variables can hold. It is to classify data.

**Examples of data types:**

* + Integer
  + String
  + Double
  + Float
  + Date
  + Character
  + Boolean
  + Long
  + Short
  + void

**Quest6:**

**Variable:**

Variable is used to label data. In simple terms it can be considered as containers that hold information (data). Variables can change values as the name specifies based on the logic. The sole purpose of variable is to label and store data in memory.

**Quest7:**

**Operator:**

Operator in a programming language is a symbol or character that tells or instructs the compiler or interpreter to perform specific mathematical, relational or logical operation and produce final result. This helps in ease of writing logic in program.

**Type of operators:**

* + Arithmetic operators
  + Relational operators
  + Logical operators

**Quest8:**

**Conditional constructs:**

Sometimes in program we require to have some restrictions or constraints based on the situation. Such situations can be handled by conditional constructs. This provides conditional flow of program. Real time scenario requires this conditional constructs as it is has to be restricted.

**Examples:**

* + If
  + If..else
  + If..else if.. else
  + switch

**Quest9:**

Switch is faster than if—else if as it directly jumps to case based on the condition, but if—else if checks for Boolean value.

Switch is readable compare to if—else if.

Switch uses primitive data type (int expression) for executing alternatives but if –else if uses Boolean condition for executing alternatives.

**Quest10:**

**Looping constructs :**

Looping constructs are used in programming for performing repetitive functions as it is required in real time. In real time we would like to perform complex and logical functions which can be done by looping constructs.

**Example:**

* + For loop
  + While loop
  + Do while loop
  + Advanced for loop

**Quest11:**

**Phases of looping:**

* + Initialization
  + Condition
  + Incrementing or decrementing

**Quest12:**

**While Loop:**

In while loop code enters the loop only when the condition is satisfied. If the condition fails, it never executes. Because condition is written at first in while loop. Initialization, condition and incrementing or decrementing is written separately.

**Do-while loop:**

In do-while loop code enters the loop even when the condition is false for the first time as the condition is written at the end of do-while loop unlike while loop. Initialization, condition and incrementing or decrementing is written separately.

**For loop:**

In for loop Initialization, condition and incrementing or decrementing is written in the same line at the starting of the loop.

**Quest13:**

**Array:**

It is a collection of related elements in an ordered arrangement. It groups data of same data-type into a single object. It stores multiple data of same data-type in a single variable.

Array is a set of continuous memory location which is used to store similar set of data inside it.

**Quest14:**

**Array in Java:**

It is a collection of related elements in an ordered arrangement. It groups data of same data-type into a single object. It stores multiple data of same data-type in a single variable.

Array is a set of continuous memory location which is used to store similar set of data inside it.

* In Java all arrays are dynamically allocated.(discussed below)
* Since arrays are objects in Java, we can find their length using member length. This is different from C/C++ where we find length using sizeof.
* A Java array variable can also be declared like other variables with [] after the data type.
* The variables in the array are ordered and each have an index beginning from 0.
* Java array can be also be used as a static field, a local variable or a method parameter.
* The **size** of an array must be specified by an int value and not long or short.
* The direct superclass of an array type is [Object](https://www.geeksforgeeks.org/object-class-in-java/).
* Every array type implements the interfaces [Cloneable](https://www.geeksforgeeks.org/marker-interface-java/) and [java.io.Serializable](https://www.geeksforgeeks.org/serialization-in-java/).

**Quest15:**

**Function return type:**

While declaring function we have to declare the return type. That function returns that value using return statement. If the function don’t return any value and it is just to perform task we can declare void as the return type.

**Quest16:**

**Use of passing parameter in the function:**

We can call a function using call-by-value and call-by-reference. These functions can be done by passing the parameters inside the function. By passing these parameters we can access the function by giving value on run time. Sometimes we pass objects as the parameters and it is known as call-by-reference. If we give value of primitive data-type then it is known as call-by-value.

**Quest17:**

**Void:**

Normally functions have return type either Integer, double, float, string, etc… but sometimes functions don’t return any value. In that situation we use void as the return type which returns nothing.

Void is the return type which returns no value. If a functions return type is void no return statement is required. Else we would want to have return value. So it is used when the function just performs task and return no value.

**Quest18:**

**Use of String[] aa:**

The main method has String[] aa as a parameter and it’s a form of standardization. Oracle does not know how many arguments a programmer will need and what types of them. For this reason, with [] aa of type String we can pass pass ‘n’ arguments to our program and convert as we need, because a string can become any primitive type in java.

**Quest20:**

**Exception handling:**

Exception handling is used to handle the runtime errors so that normal flow of the application can be maintained. Exception is an event that disrupts the normal flow of the program. It is thrown at runtime.

It is the mechanism to handle runtime errors such as classNotFound, IO, SQL, etc…

Advantage of exception handling is to maintain the normal flow of the application even when the exception is thrown.

**Quest21:**

Exception handling is to maintain the normal flow of the application even when the exception is thrown. In the case of exception we don’t want our program to stop abruptly and sometimes exceptions might occur at a least important part. So we don’t want our main part of our program to stop abruptly. To avoid this situation we use exception handling.

**Quest22:**

We can handle exceptions using try and catch block. Inside the try block we have write the code where the exception might occur. In the catch block we can handle that exception.

We can also use throw and throws keyword to handle exception. In this way we can handle exception in program.

**Quest23:**

**Constructor:**

Constructor in java is similar to method or function that’s called when an instance of an object is created for that class. It does not required to invoke or called separately. Constructor doesn’t have any return type and has the same name as that of the class.

**Quest24:**

Constructor is used like a method or function in java. Constructor is also referred as the special method in java. It is called automatically when instance of the class is created. It is not required to call the method separately.

**Quest25:**

**Parameterized constructor:**

If we have parameters in the constructor then it is known as parameterized constructor.

We can call a constructor using call-by-value and call-by-reference also. These functions can be done by passing the parameters inside the function. By passing these parameters we can access the function by giving value on run time. Sometimes we pass objects as the parameters and it is known as call-by-reference. If we give value of primitive data-type then it is known as call-by-value.

**Quest26:**

**Rules to be followed in using constructor:**

* + Constructors are not methods and they don’t have any return type
  + Constructor name should match with class name
  + Constructor can use any access specifier
  + Constructor overriding is not possible
  + Interfaces don’t have constructors

**Quest27:**

We can use the base class constructor in child class using super() keyword. If we have parameterized constructor in base class then we can use super(parameter list) to call the base class constructor.

**Quest28:**

Constructors cannot be inherited. So we have to call them using super() keyword. So the flow of constructor execution will be there only when we call them as it is not inherited generally. So without calling constructor it is not executed in the flow.

**Quest29:**

**Overriding:**

Overriding is used in inheritance. When we inherit a method from parent class sometimes we would like to change its behavior that’s specific to the subclass type based on the requirement. In simpler terms overriding means to override the functionality of an existing method.

This can be implemented by having the same method name with same parameters as that of parent class method but different functionality. By this we can implement overriding.

**Quest30:**

**Overloading:**

Overloading is used in inheritance. It allows a class to have more than one method having the same name but different argument lists.

We can overload method in the following ways:

* + - By changing the number of parameters
    - By changing the data type of parameters
    - By changing the sequence of parameters

**Quest31:**

**Inheritance:**

Inheritance is used in java by which one class is allowed to inherit the features of another class. It is used for code reusability.

In program we implement inheritance by using the keyword extends.

**Quest32:**

Class is inherited using extends keyword while interface is inherited using implements keyword.

**Quest33:**

**Interface:**

An interface in the java programming language is an abstract type that is used to specify a behavior that classes must implement. Interfaces use interface keyword and may contain only method signature and constant declarations. Method can be only of abstract type. Interfaces cannot be instantiated. Interface supports multiple inheritances.

|  |  |
| --- | --- |
| Abstract class | Interface |
| It can have both abstract and non abstract methods. | It can have only abstract methods |
| It doesn’t support multiple inheritance | It supports multiple inheritance |
| It can have final, non-final, static, and non-static variables | It has only static and final variables |
| It can implement interface | It can’t implement abstract class |
| Abstract keyword is used to declare abstract class | Interface keyword is used to declare interface |

**Quest34:**

**Framework:**

A framework is a platform for developing applications. It provides a foundation on which software developers can build programs for a specific platform. It provides set of classes and interface.

**Quest35:**

**Collections:**

Collections in java are a framework that helps to store and manipulate the group of data. All the operations that we perform on data such as searching , sorting, insertion, manipulation, deletion, etc. can be performed by java collections.

Java collections are set, list, queue, deque, etc.

**Quest36:**

**Collection framework:**

Java collections framework is like a library which contains set of classes and interfaces that helps to perform collections task.

**Quest37:**

**ArrayList:**

ArrayList class extends AbstractList and implements the List interface. ArrayList support dynamic arrays that can grow as needed. Normally arrays are of a fixed length. After arrays are created they cannot grow or shrink. To overcome this we can use ArrayList. When size of ArrayList is exceeded, the collection is automatically enlarged, when objects are removed, then it is shrunk.

**Quest38:**

**Iterator:**

Iterator is used to traverse through the elements in a collection and it is used to display or remove elements in collections. It also traverse in bidirectional. Iterator implements either iterator or the listiterator interface.

**Quest39:**

**Characteristics of ArrayList are:**

* Java ArrayList class can contain duplicate elements.
* Java ArrayList class maintains insertion order.
* Java ArrayList class is non synchronized.
* Java ArrayList allows random access because array works at the index basis.
* In Java ArrayList class, manipulation is slow because a lot of shifting needs to be occurred if any element is removed from the array list.

**Quest40:**

**Advantages of ArrayList:**

It supports dynamic array. That size can be varied during the runtime.

Arraylist allows random access because array works at the index basis.

ArrayList can have duplicate values

ArrayList maintains insertion order

**Quest42:**

**LinkedList:**

It is also used to store multiple objects and perform tasks on those. Linkedlist uses doubly linked list to store the elements. It can store data in both the direction.

**Quest43:**

**ListIterator:**

It is used to iterate elements one by one from a list. It extends Iterator interface. It supports create, read, update and delete operations. It supports both forward and backward direction iterations. It is a bi-directional iterator.

**Quest44:**

**Characteristics of LinkedList :**

* Java LinkedList class can contain duplicate elements.
* Java LinkedList class maintains insertion order.
* Java LinkedList class is non synchronized.
* In Java LinkedList class, manipulation is fast because no shifting needs to be occurred.
* Java LinkedList class can be used as list, stack or queue.

**Quest45:**

**Advantages of LinkedList:**

Manipulations is fast

It can be used as list, stack or queue

It uses doubly linked list to store the elements

We can add or remove elements from both the sides

**Quest46:**

**LinkedList in real life software:**

Web history page

Low level memory management

Hashtables

**Quest47:**

|  |  |
| --- | --- |
| **arraylist** | **linkedlist** |
| ArrayList internally uses **dynamic array** to store the elements | LinkedList internally uses **doubly linked list** to store the elements. |
| Manipulation with ArrayList is **slow** because it internally uses array. If any element is removed from the array, all the bits are shifted in memory. | Manipulation with LinkedList is **faster** than ArrayList because it uses doubly linked list so no bit shifting is required in memory. |
| ArrayList class can **act as a list** only because it implements List only. | LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces. |
| ArrayList is**better for storing and accessing**data. | LinkedList is **better for manipulating** data. |

**Quest48:**

**Generic class:**

Generic class helps in giving type safety that allows compile time checking of types. It helps in storing specific type of objects. It provides type safety, eliminates type casting and enables compile time checking.

**Quest49:**

**Benefits of generic class:**

Type safety

Eliminates type casting

Provides compile time checking

**Quest51:**

**HashSet:**

HashSet is used for storing the collection of object in a hash table. It stores by using the mechanism of hashing.

**Quest52:**

List can contain duplicate elements whereas set contains unique elements only.

**Quest53:**

* + Hashset class inherits AbstractSet class and implements Set interface
  + HashSet stores the elements by using a mechanism called hashing
  + HashSet contains unique elements only

**Quest56:**

Constructor is invoked when the instance of the class is created. It need not be called separately. So when we want that task to be performed when the class is called without calling that method then we can use constructor.

**Quest57:**

* + Constructors are not methods and they don’t have any return type
  + Constructor name should match with class name
  + Constructor can use any access specifier
  + Constructor overriding is not possible
  + Interfaces don’t have constructors

**Quest58:**

|  |  |
| --- | --- |
| **constructor** | **Method** |
| Name of the constructor must be same as that of the class | Name can be anything |
| It doesn’t have any return type. | It has return type |
| It doesn’t require to call separately | It has to be called separately |
| It is not inherited | It is inherited |

**Quest59:**

**Inheritance:**

Inheritance is used in java by which one class is allowed to inherit the features of another class. It is used for code reusability.

In program we implement inheritance by using the keyword extends.

**Quest60:**

In program we implement inheritance by using the keyword extends.

**Quest61:**

Access specifier governs the access of the variables and methods by other subclasses or methods. Private method or variable cannot be accessed by the subclass. Other access specifier have no significance as they can be accessed by the sub class.

**Quest62:**

Multiple inheritance is not supported in java as it creates ambiguity. Since overriding and overloading is allowable multiple inheritance causes ambiguity problem (i.e. diamond problem). Hence it is not used in java.

**Quest63:**

Constructors are not inherited in java. It has to be called using the super keyword.

This keyword represents the current instance of the object.

Super keyword is used to call the parent class constructors, and fields.

**Quest64:**

**Polymorphism:**

Polymorphism is the ability of an object to take on many forms. It is done by method overloading and method overriding.

**Benefits of polymorphism:**

It is used t change the method according to the scenario

We can have more parameters as required

**Quest65:**

Polymorphism is implemented by

Method overloading

Method overriding

**Quest66:**

Method overloading is changing the parameters of the method. It is done by increasing the parameters, or changing the data-type, or changing the sequence.

While implementing method overloading only parameters are changed but the functionality is not changed inside the method whereas in method overriding functionality is changed but not the parameters.

**Quest67:**

Method overriding is changing the functionality of the method. It is done by changing the functionality of the method inside it. But parameters and other things are not changed.

While implementing method overloading only parameters are changed but the functionality is not changed inside the method whereas in method overriding functionality is changed but not the parameters.

**Quest68:**

**Final keyword:**

Final keyword can be used for variable, method and class. When the final keyword is used for the variable its value cannot be changed. When the final keyword is use for the method then the method cannot be overridden. It restricts the user by not allowing them to change the value.