**Assignment-7**

**1)What is oops?**

1)Object means a real-world entity such as a pen, chair, table, computer, watch, etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

* [Object](https://www.javatpoint.com/object-and-class-in-java)
* Class
* [Inheritance](https://www.javatpoint.com/inheritance-in-java)
* [Polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java)
* [Abstraction](https://www.javatpoint.com/abstract-class-in-java)
* [Encapsulation](https://www.javatpoint.com/encapsulation)

Object:Any entity that has state and behaviour is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical. An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory.

Class:Collection of objects is called class. It is a logical entity. A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance:When one object acquires all the properties and behaviours of a parent object, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

Polymorphism:If one task is performed in different ways, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

Abstraction: Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing. In Java, we use abstract class and interface to achieve abstraction.

### Encapsulation:Binding code and data together into a single unit are known as encapsulation.

**2)What is classes in java?**

**2)** Instances of the class Class represent classes and interfaces in a running Java application. An enum is a kind of class and an annotation is a kind of interface. Every array also belongs to a class that is reflected as a Class object that is shared by all arrays with the same element type and number of dimensions. The primitive Java types (boolean, byte, char, short, int, long, float, and double), and the keyword void are also represented as Class objects.

Class has no public constructor. Instead Class objects are constructed automatically by the Java Virtual Machine as classes are loaded and by calls to the defineClass method in the class loader.

**3)What is structure in java?**

**3)** The standard mapping in the Java programming language for an SQL structured type. A Struct object contains a value for each attribute of the SQL structured type that it represents. By default, an instance of Struct is valid as long as the application has a reference to it.

All methods on the Struct interface must be fully implemented if the JDBC driver supports the data type.

**4)What is type casting in java?**

**4**) Java data type casting comes with 3 flavors.

* Implicit casting
* Explicit casting
* Boolean casting.

1. Implicit casting (widening conversion)

|  |  |
| --- | --- |
|  |  |

A data type of lower size (occupying less memory) is assigned to a data type of higher size. This is done implicitly by the JVM. The lower size is widened to higher size. This is also named as automatic type conversion.

2. Explicit casting (narrowing conversion)

A data type of higher size (occupying more memory) cannot be assigned to a data type of lower size. This is not done implicitly by the JVM and requires explicit casting; a casting operation to be performed by the programmer. The higher size is narrowed to lower size.

3. Boolean casting

A Boolean value cannot be assigned to any other data type. Except Boolean, all the remaining 7 data types can be assigned to one another either implicitly or explicitly; but Boolean cannot. We say, Boolean is incompatible for conversion. Maximum we can assign a Boolean value to another Boolean.

**5)What is recycler View?**

**5)** The [Recycler View](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.html) widget is a more advanced and flexible version of View. In the [Recycler View](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.html) model, several different components work together to display your data. The overall container for your user interface is a [Recycler View](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.html) object that you add to your layout. The [Recycler View](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.html) fills itself with views provided by a layout manager that you provide. You can use one of our standard layout managers (such as [LinearLayoutManager](https://developer.android.com/reference/androidx/recyclerview/widget/LinearLayoutManager.html) or Grid Layout), or implement your own.

The views in the list are represented by view holder objects. These objects are instances of a class you define by extending [Recycler View. View Holder](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.ViewHolder.html). Each view holder is in charge of displaying a single item with a view. For example, if your list shows music collection, each view holder might represent a single album. The [Recycler View](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.html) creates only as many view holders as are needed to display the on-screen portion of the dynamic content, plus a few extra. As the user scrolls through the list, the [Recycler View](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.html) takes the off-screen views and rebinds them to the data which is scrolling onto the screen.

The view holder objects are managed by an adapter, which you create by extending [Recycler View. Adapter](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.Adapter.html). The adapter creates view holders as needed. The adapter also binds the view holders to their data. It does this by assigning the view holder to a position, and calling the adapter's [on Bind View Holder()](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.Adapter.html#onBindViewHolder(VH,%20int)) method. That method uses the view holder's position to determine what the contents should be, based on its list position.

This [Recycler View](https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.html) model does a lot of optimization work so you don't have to:

**6)What is raw handler?**

6)A Handler allows you to send and process [Message](https://developer.android.com/reference/android/os/Message.html) and Runnable objects associated with a thread's [Message Queue](https://developer.android.com/reference/android/os/MessageQueue.html). Each Handler instance is associated with a single thread and that thread's message queue. When you create a new Handler, it is bound to the thread / message queue of the thread that is creating it -- from that point on, it will deliver messages and runnable to that message queue and execute them as they come out of the message queue.There are two main uses for a Handler: (1) to schedule messages and runnable to be executed at some point in the future; and (2) to enqueue an action to be performed on a different thread than your own.