Day-8

Agenda

Interface
OOP’S
Collections

Interface

Any service requirement specification (SRS) is considered as Interface

From the client point of view an Interface defines the set of service what is expecting

From the service provider point of view an Interface defines the set of service what is offering

Hence, An Interface considered as contract between client & service provider

Advantage of Interface

We can achieve security because we are not highlighting our internal implementation

Enhancement will become very easy because without effecting outside person we can change our internal implementation .

Declaration of Interface

We can declare the interface by using interface keyword, we can implement an interface by using implements keywords.

Extends Vs Implements

Interface

Extends Vs Implements

A class can extends only one class at a time .

A class can implements any no of interface at a time

A class can extends a class and implement any no of interface at a time

Interface Method

Every method in interface by default default, public, abstract

Interface Variables

Every variables in interface by default public, static, final

Interface Naming conflict

Case -1

If two interface contains a method with same signature & same return type in the implementation class ,

we can provide implementation for only one method

Case -2

If two interfaces contains a method with same name but different argument then, in the implementation class

We have to provide implementation for both methods & hence methods are considered as overloaded methods

Case-3

If two interface contains a method with same signature but different return types than it is impossible to implement both interface at at time .

Marker Interface

If an interface would not contain any method & by implementing that interface if our object will get ability

Such types of interface are called marker interface or tag interface or ability interface.

Ex : Serializable , clonable , Random Access etc…

Interface

Interface	Abstract Class
If we don’t know any thing about implementation just we have the. Requirement than we should go for interface	If we are talking about implementation but not completely then we should go for abstract class
Every method present inside the interface is by default public & abstract	Every method present inside abstract class need not be public & abstract . We can take concrete method also
The following modifiers are not allowed for interact method . Strictfp, protected, static, native, private , final , synchronized .	There are no restriction for abstract class method modifiers . We can use any modifiers
Every variable present inside interface is public , static , final , by default .	Abstract variable need not be public , final , static
For the interface variables we can not declare the following modifiers : private , protucted, transient, volatile	**There are no restriction for abstract class variable modifiers .**
For the interface variables compulsory we should perform initialization at the time of declaration only	**For the abstract class variables there is no restriction like performing initialization**
Inside interface we can not take constructor	**Inside abstract class we can take constructor .**

OOP’s Concept

Abstraction

Encapsulation

Polymorphism

Inheritance

OOP’s Concept

Abstraction

Hiding internal implementation details & just highlights the set of service what we are offering is called “Abstraction”

Encapsulation

Encapsulation data & corresponding methods in to a single modules is called “encapsulation”

Polymorphism

Where the nature of object decides at run time is called polymorphism .

- **Run Time**
- **Compile Time**

OOP’s Concept

Polymorphism

one object is responsible to perform multiple task at run time is called polymorphism .

There are Two types of polymorphism

- Run Time or Method Overriding
- Compile Time or Method Overloading

What is Covarient Return type

Child return type need not to be same as parent method returns type. Where child classes also allowed .

This is only applicable for object type not for primitive type

Ex- Object -> String , Object -> Integer , Number -> Integer etc…

OOP’s Concept

Compile Time or Method Overloading

Two method are said to be overloaded iff method names are same but arguments are different

Run Time or Method Overriding

Two method are said to be overridden iff method name & signature must be matched .

In overriding return type must be matched , but this rules is applied until 1.4 v. But from 1.5 v onwards Covarient returns types are allowed .

We can not override the final & private type method

We can not override the static method , it is called method hiding .

Difference between Overloading & Overriding :

Property	Overloading	Overriding
1. Method Name	Must be same	Must be same
2. Arguments	Must be different	Must be same
3. Method Signature	No restriction	Must be same
4. Private , Static , & final	Can be overloaded	Can not be overloaded
5. Access Modifers	No restriction	We can not decrease the scope
6.Method resolution	Always take care by the compiler based on reference type	Always take care by the JVM based on runtime
7. Also known as	Compile time , static polymorphism , early binding	Run time , Dynamic polymorphism , late binding .

OOP’s Concept

Inheritance

It is also known as “Is-A relationship ”

By using extends keyword we can implement Is-A Relationship

The main advantage of Is-A relationship is reusability of the code

Has-A Relationship

This is also known as composite or aggregation

There is no specific keywords to implement Has-A relationship . Mostly we use new keywords

The main dis-advantage of Has-A relationship is to increase the dependency between class

Break

Will be back at 09:20 PM IST