Java Program: Course 3: Plan

STACK ROUTE

Program — Build job-ready skills

- Java Foundation Program

Courses — Build competencies

- Problem Solving and Computational Thinking using Java
- Introduction to Programming in Java
- **Object Oriented Programming in Java**
- Data Structures and Algorithms in Java
- Advanced Programming in Java
- Relational Database Management Systems (RDBMS) Concepts and SQL
- Java Programming Project

Learning Sprints — Perform specific tasks

Schedule

Learning Sprints	Schedule
Think and model real world as Objects	Day 1
Introduction to Encapsulation and Data Abstraction	Day 1
Implement Encapsulation and Data Abstraction in your Java program	Day 2
Create new class that inherits from parent class	Day 3
Introduction to polymorphism	Day 3
Implement Inheritance and Polymorphism using Abstract class and Interface	**Day 4**
Implementing Wrapper Class and Inner Class	**Day 4**
Consolidate OO programming in Java and Write Clean Code	Day 5

Think and Tell

Different banks use different methods to calculate the interest rate. However, it is important for the banks to provide a concrete definition for interest calculation.

How can we implement this programmatically?

Gaming Console

In a game console, each game offers functionalities to play the game, compute and display the score, and exit the game. Some of these functionalities are common and some vary from game to game.

How can we represent the common functionalities of an application and reuse them at multiple places?

Let Us Discuss

How can we create a class structure for both the scenarios?

Implement Inheritance Using Abstract Class and Interface

Learning Objectives

- Define abstraction
- List different ways to achieve abstraction
- Implement abstraction by using an abstract class
- Apply abstraction by using an interface

Abstraction

- In object-oriented programming, we hide the unnecessary information and show only necessary information

- Abstraction mainly focuses on what an object does instead of how it does it

- Here, only the functionality is provided to the user. Implementation of the functionality remains hidden from the user

- Abstraction can be achieved by using:
  - Abstract classes
  - Interfaces

Inheritance Using an Abstract Class

- An abstract class contains one or more abstract methods. It cannot be instantiated but it can be inherited by other classes by using the extends keyword

- When an abstract class is inherited by a subclass, the subclass must provide the implementation of all the abstract methods defined in the abstract class

- Syntax of declaring an abstract class:

```
<access_specifier> abstract class <abstract_class_name>
{
   //variables
   // abstract and concrete methods
}
```

Abstract Methods

- An abstract method does not have its body
- We use the same abstract keyword to create abstract methods
- Abstract methods are mostly declared where two or more subclasses are also doing the same thing in different ways through different implementations
- To use an abstract class, you have to inherit it from another child class and provide implementation of the abstract method in it
- Syntax of declaring abstract method

```
abstract return-type methodname(parameter-list);
```

Important Points for Abstract Classes

- We use the abstract keyword to create abstract classes and methods

- An abstract method does not have any implementation and we cannot create objects of an abstract class

- To implement its features, we inherit subclasses from it and create objects of the subclass

- A subclass must override all abstract methods of an abstract class. However, if the subclass is declared abstract, it's not mandatory to override abstract methods

- We can access the static attributes and methods of an abstract class using the reference of the abstract class

Interactive Demo

Games, in a game console offer functionalities to play the game, compute and display the score, and exit the game. Some functionalities, such as display the score and exit the game, are the common is all the consoles, but there are some functionalities, such as play the game, compute the score, etc., vary for every game in the console.

For example, a badminton game would have a different mechanism than a table tennis game to compute the score. However, the manner in which scores are displayed in both the games might be similar.

Write a program in Java to achieve the above objective.

Interfaces

- Interfaces contain a set of abstract methods and static data members
- By default, all the methods specified in an interface are public and abstract. All the variables are public, static, and final
- It is used to achieve abstraction, polymorphism, and multiple inheritance
- Syntax for defining an interface

```
interface <interfacename>
{
       //interface body
       static final <data type> <variablename>;
       public return-type methodname(parameter);
}
```

Using Interface to Implement Inheritance

- An interface can be implemented in one or more classes
- Interface can be implemented in a class by using the following syntax:

```
class <class_name> implements [interfacename]
{
       //Defining the method declared in the interface.
public return-type methodname(parameter-list)
{
       //Body of the method
}
}
```

- An interface can also extend an interface, by using the following syntax:

```
interface <interface_name> extends <interface1, interface2
, …interfaceN>
```

Points to Remember

- An interface cannot be instantiated
- It cannot have a constructor
- It can extend itself to multiple interfaces
- It contains only abstract methods, static and final variables

Interactive Demo

A mobile phone has various functionalities. Some functionalities, such as to make, pick or disconnect a call, remain the same for all mobile phones while some functionalities might vary from one phone to another.

Write a program in Java to achieve the above objective of representing these functionalities by using interfaces for various electronic devices, such as a computer, a laptop, etc.

Super and Final Keywords

- Super keyword refers to superclass objects. It is used to call superclass methods and access the superclass constructor. It helps in eliminating the confusion between the parent and child class methods that have the same name

- Final keyword is a non-access modifier used for classes, attributes, and methods, which make them non-changeable. It is useful when you want a constant variable which always stores the same value

Key Takeaways

- Abstraction and its uses
- Different ways to achieve abstraction
- Use abstract classes and interfaces to Implement abstraction
- Super and Final Keywords

Thank you!