

Module 7: Inheritance and Interfaces

Module 8: More on Classes





Review

- **Arrays is special data store that can hold several items of a single data type in contiguous memory locations**
- **String is a class for immutable text data**
- **StringBuilder store a grow able and flexible string.**
- **StringTokenizer is a class for break a string into subunits**



Review

- **Java package is a group of related classes and interfaces organized as one unit.**
- **Packages can be classified as predefined packages and user-defined packages.**
- **The package statement, if it is added, must be the first line in Java code.**
- **Access modifiers control the access and visibility of a class and class members.**
- **Field and method modifiers are keywords used to identify fields or methods that need to be declared for controlling access to users.**



Objective of module 7

- **Inheritance**
- **Overloading of methods**
- **Using abstract keyword**
- **Using final keywords**
- **Interfaces**



Concept of inheritance

- **The process, whereby characteristics and behavior are transmitted from a parent to a child entity, is called inheritance**
- **Inheritance aims to avoid creating characteristics and behavior that already exist and instead reuses the existing ones and builds on them to create new entities**



Feature and terminology

- **Inheritance enables you to define a very general class first, and then define more specialized classes later by simply adding some new details to the general class definition**
- **The class which inherits the fields and methods is called sub-class**
- **The class from which the sub-class inherit the fields and methods is called superclass**



Subclassing

- **The extends keyword is used to create sub-class**
- **A class can be directly derived from only one class**
- **If a class does not have any superclass, then it is implicitly derived from Object class**
- **Unlike other members, constructor cannot be inherited**



Overriding methods

- **When a sub-class defines a new method having the same signature as the superclass method, then the process is called overriding**
- **Method overriding rules:**
 - Same signature
 - Cannot have weaker access specifier
- **The super keyword**
 - Accessing superclass methods
 - Accessing superclass constructor



Concept of overloading

- **The ability of a class to define several methods with the same name**
- **Different number of parameters and same data types**
- **Same number of parameters and different data types**
- **Similar method, constructor can be overloaded, the same rules**



Using the this keyword

- **Is used in the constructor or instance method to refer to the current object**
- **Is used to access members that are presently shadowed or hidden by local variable with the same name**



Using the abstract keyword

- **Abstract method**

```
abstract returnType methodName(parametersList);
```

- **Abstract class**

- **Cannot be instantiated but can be inherited**
- **The sub-class must implement abstract methods declare in base class**
- **Otherwise it must be declared as abstract**



Using the final keyword

■ Final variables

- ✓ Constants
- ✓ Initialized at declaration
- ✓ Value cannot be modified

■ Final classes

■ Final methods

- ✓ Cannot be overridden
- ✓ Abstract keyword cannot be used
- ✓ Implicit final methods

- ✓ Cannot be subclassed
- ✓ May or may not have final methods
- ✓ Final classes can be instantiated



Interface

- **Provide a workaround by allowing classes to inherit one or more interfaces in addition to inheriting a class**
- **Reference type**
- **Cannot be instantiated**
- **Classes implement interfaces**
- **Interface can extend an interface**



Summary of module 7

Inheritance

Through Inheritance the characteristics and behavior are transmitted from a parent entity to a child entity. The inheriting class is called the subclass and the inherited class is called the superclass. Every Java class has exactly one superclass which can be either a user-defined class or the `Object` class.

In Overriding, a subclass defines a new instance method having the same signature as the superclass method. The `super` keyword allows an overridden method of a superclass to be invoked from the subclass method. It can also be used to access instance variables of the superclass. Since constructors are not inherited, `super()` is used to invoke the constructor of a superclass in a subclass constructor.

Overloading of Methods

Method overloading is the ability of a class to define several methods with the same name. A method can be overloaded only by having either different number of parameters and same data types or same number of parameters and different data types. Similar to methods, constructors can be overloaded. The same rules that apply for method overloading also apply to constructors.

The `this` keyword in Java is used within a constructor or instance method to refer to the current object in memory.



Summary of module 7 (cont.)

Using abstract Keyword X

When a method has only declaration and no action statements in the body, it is called an abstract method. A Java class containing one or more abstract methods is known as an abstract class.

Using final Keyword X

A final variable is declared with the `final` keyword and assigned a value at the time of declaration. A compile time error is raised if a final variable is reassigned a value in a program after its initial declaration. To prevent a method from being overridden or hidden in a Java sub class, it is declared as `final`. A class that cannot be sub classed is called a final class in Java. The `final` keyword is applied to the class declaration to achieve this.

Interfaces X

An interface is defined as a reference type and is similar to a class except that it has only final variables and abstract method signatures. An interface cannot be instantiated. It can be implemented by other classes or extended by other interfaces.



Objective of module 8

- **Class variables**
- **Nested class**



Class variables

- **Are declared using the static keyword**
- **Once the value is modified, all instances of the class are updated to share the same value**
- **Can also be manipulated without creating an instance of the class**
- **Are referenced by the class name itself**



Static methods

- **Also known as class methods, do not have reference to any instance variable in a class, used to access class variables and methods**
- **The keyword this cannot be used inside a static method**
- **Can be accessed by using a class name or an instance name**



Static initializer

- **Is a block of code embraced in curly braces**
- **It is preceded by the static keyword**
- **It initializes the static variables in class**
- **The static initialization block can reference only class variables that have been declared before it**



Nested class

- **A class defined within another class**
- **It can have access to members of the outer or enclosing class even if the members are declared private**
- **It can be used for:**
 - **Allow logical grouping of classes**
 - **Increases encapsulation**
 - **More maintainable code**



Different type of nested class

- **Member classes or non-static nested classes**
- **Local classes**
- **Anonymous classes**
- **Static nested classes**



Summary of module 8

Class Variables



Class Variable is allocated memory initially when the class is loaded. Only one variable is used for all the objects of the class. Static methods use class variables, not any instance variables of the class.

Nested Class



A nested class is defined within another class. A nested class is also known as inner class. Nested classes can be static or non-static. Non-static nested classes are also called member classes. A nested class is defined within another class. The different types of nested classes are Member classes or non-static nested classes, Local classes, Anonymous classes and Static Nested classes.