Java Programming Tutorial

By
Ujjwal Biswas

Content

- Classes and Objects
- Inheritance
- Packages & Interfaces
- Exception Handling
- Utility Classes & Interfaces
- Event Handling Using Swing
- I/O Streams
- Networking

The Java Buzzwords

- Simple
  - Small language [large libraries]
  - Small interpreter (40 k), but large runtime libraries (175 k)
- Object-Oriented and software technologies
  - Supports encapsulation, inheritance, abstraction, and polymorphism.
  - structured error-handling
  - garbage collection
- Distributed
  - Libraries for network programming
  - Remote Method Invocation
- Architecture neutral
  - Java Bytecodes are interpreted by the JVM.

The Java Buzzwords (Contd.).

- Secure
  - Difficult to break Java security mechanisms
  - Java Bytecode verification
  - Signed Applets.
- Portable
  - Primitive data type sizes and their arithmetic behaviour specified by the language
  - Libraries define portable interfaces
- Multithreaded
  - Threads are easy to create and use
- Dynamic
  - Finding Runtime Type Information is easy

Class, Object and Encapsulation

- A class is a template for an object.
  - Class defines a new data type.
  - This new type can be used to create object of that type.
- A class has a class-name, a set of attributes and a set of services or actions.
- The object is the instance of the class.
- Encapsulation is the ability of an object to be a container/capsule
  - related **properties** (ie. data variables) and **methods** (ie. functions).
- Encapsulation makes it easy to maintain and modify code.
  - If method signature remains unchanged, client code is not affected with any internal change in method

Function Overloading and Constructor

- Identical name functions are said to be **overloaded**.
- Overloaded function calls are resolved using function signatures which constitutes the function name, the number, order and the data type of the arguments.
- A constructor function has the same name as the class name.
  - A class can contain more than one constructor.
  - Facilitates multiple ways of initializing an object
  - Constructors can be overloaded.

The final Keyword

- Many programming languages have a way of telling the compiler that a piece of data is “constant”.
- A constant is useful for two reasons:
  - It can be a compile-time constant that will never change.
  - It can be a value initialized at runtime that you don’t want changed.
- A variable can be declared as **final**.
- Must initialize a **final** variable when it is declared
  - **final float PI = 3.142857**;
- This prevents its contents from being modified.
- **Thus a final variable is essentially a named constant.**

Access Specifiers

- Access specifiers help implement:
  - Encapsulation by hiding implementation-level details in a class
  - Abstraction by exposing only the interface of the class to the external world
- The **private** access specifier is generally used to encapsulate or hide the member data in the class.
- The **protected** is used by the class itself, also by the subclasses of the class and by all the classes in the same package.
- The **public** access specifier is used to expose the member functions as interfaces to the outside world.

Inheritance

- Inheritance is one of OOP concepts it allows for the creation of hierarchical classifications.
- Using inheritance, you can create a general class at the top.
- This class may then be inherited by other, more specific classes.
- Each of these classes will add only those attributes and behaviours that are unique to it.

Generalization/Specialization

- In keeping with Java terminology, a class that is inherited is referred to as a **superclass**.
- The class that does the inheriting is referred to as the **subclass**.
- **Each instance of a subclass includes all the members of the superclass.**
- **The subclass inherits all the properties of its superclass.**

What is an Interface?

- **Definition:** An interface is a named collection of method declarations (without implementations)
  - An interface can also include constant declarations.
  - Interfaces are an integral part of Java.
  - **An interface is a complex data type similar to class in Java**.
  - An interface is syntactically similar to an abstract class.
  - An interface is a collection of abstract methods and final variables.

Types of Inheritance

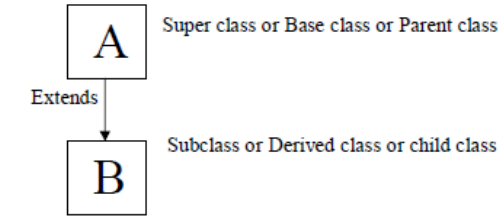


Fig: Single Inheritance

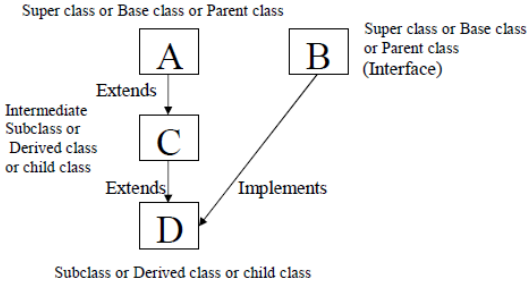
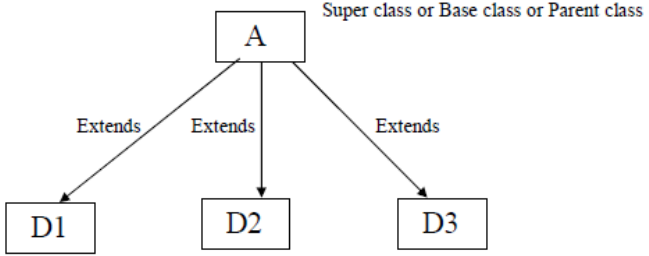


Fig: Hybrid Inheritance



D1, D2, D3 are the Subclass or Derived class or child class of A.

Fig: Hierarchical Inheritance

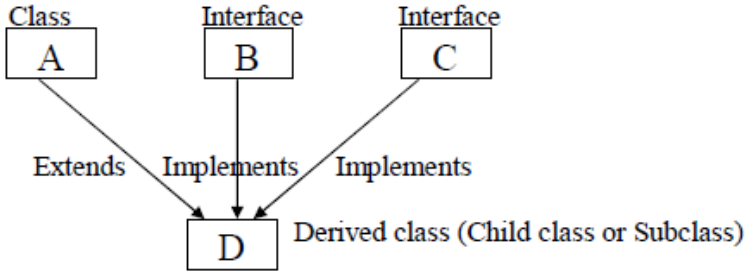


Fig: Multiple Inheritance

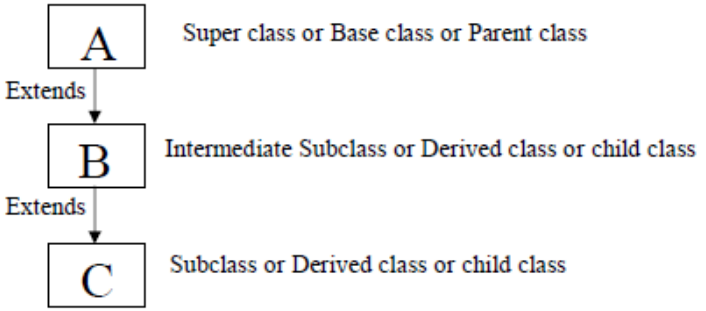


Fig: Multilevel Inheritance

Access Control

Specifier	Accessibility
private	Accessible in the same class only
protected	Subclasses and non-subclasses in the same package, and subclasses in other packages
No-specifier (default access)	Subclasses and non-subclasses in the same package
public	Subclasses and non-subclasses in the same package, as well as subclasses and non-subclasses in other packages. In other words, total visibility

A Package Example

Event Handling and GUI based programming

- In console-based programs, the program:
  - may prompt the user for some input
  - processes the user input and displays the result
- In a GUI-based program, the user initiates the interaction with the program through GUI events such as a button click.
- In a GUI environment, the user drives the program.

Event Handling and GUI based programming(Contd.).

- Whenever a user interacts with these GUI controls:
  - some event is generated
  - some action implicitly takes place that validates or adds functionality to the event
- This type of programming is called **event-driven** programming, where the program is driven by the events.
- Events are best used in GUI-based programs.

Example of an Event

- **MouseEvent** class is a subclass of the **InputEvent** class.
- A mouse event is generated when the mouse is pressed, released, clicked, entered, exited etc

Thank You