Single, Multilevel, and Multiple Inheritance

Single Inheritance: A class inherits from only one superclass.

Example:

```
class Animal {
   void eat() {
      System.out.println("Eating...");
   }
}

class Dog extends Animal {
   void bark() {
      System.out.println("Barking...");
   }
}
```

Multilevel Inheritance: A class is derived from a class that is also derived from another class.

Example:

```
class Animal {
   void eat() {
      System.out.println("Eating...");
   }
}

class Dog extends Animal {
   void bark() {
      System.out.println("Barking...");
   }
}
```

```
class Puppy extends Dog {
   void weep() {
      System.out.println("Weeping...");
   }
}
```

Multiple Inheritance: Java does not support multiple inheritance directly (a class cannot inherit from more than one class) to avoid complexity and simplify the language. However, it can be achieved using interfaces.

Example:

```
interface Animal {
   void eat();
}

interface Pet {
   void play();
}

class Dog implements Animal, Pet {
   public void eat() {
      System.out.println("Eating...");
   }

   public void play() {
      System.out.println("Playing...");
   }
}
```

1. **Using `super` for Parent Variables**

```
class Parent {
```

```
    String message = "Hello from Parent!";
}

class Child extends Parent {
    String message = "Hello from Child!";

    void printMessages() {
        System.out.println(message);        // Accessing Child's message
        System.out.println(super.message);  // Accessing Parent's message
using super
    }

    public static void main(String[] args) {
        Child child = new Child();
        child.printMessages();
    }
}
```

2. Using `super` for Parent Methods

```
class Parent {
    void displayMessage() {
        System.out.println("Message from Parent class.");
    }
}

class Child extends Parent {
    void displayMessage() {
        System.out.println("Message from Child class.");
    }

    void showMessage() {
        displayMessage();
        super.displayMessage();
    }
```

```
    public static void main(String[] args) {
        Child child = new Child();
        child.showMessage();
    }
}
```

- **Final Variables**: Declared using `final`, their value cannot change once initialized.
- **Final Methods:** Declared using `final`, they cannot be overridden by subclasses.
- **Final Classes:** Declared using `final`, they cannot be extended by other classes

Abstract class	Interface
1)Abstractclasscan**haveabstract andnon-abstract**methods.	Interfacecanhave**onlyabstract**methods.Since Java 8, it can have **default and static methods**also.
2) Abstract class**doesn't support multipleinheritance**.	Interface **supportsmultipleinheritance**.
3)Abstractclass**canhavefinal,non-final,staticandnon-staticvariables**.	Interfacehas**onlystaticandfinalvariables**.
4) Abstract class**can provide the implementationofinterface**.	Interface**can'tprovidetheimplementationof abstractclass**.
5)The**abstractkeyword**isusedto declareabstractclass.	The**interface keyword**is used to declare interface.

6)**Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

1. Using Built-in Packages

Java comes with a set of built-in packages that provide various functionalities. Here's a simple example using the `java.util` package, which includes utility classes like `Scanner` for input:

```
import java.util.Scanner; // Importing Scanner class from
java.util package

public class PackageExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = scanner.nextLine();

        System.out.println("Hello, " + name + "! Welcome to Java
packages.");

        scanner.close();
    }
}
```

2. User-Defined Packages

You can create your own packages to organize your classes and make them reusable across different projects. Here's a simple example of how to create and use a user-defined package:

```
//savebyA.java package pack;
 publicclassA{
public void msg(){
System.out.println("Hello");
}
}
//save by B.java
packagemypack;
 class B{
publicstaticvoid main(Stringargs[]){
pack.Aobj=newpack.A();
```

```
obj.msg();
}
}
```