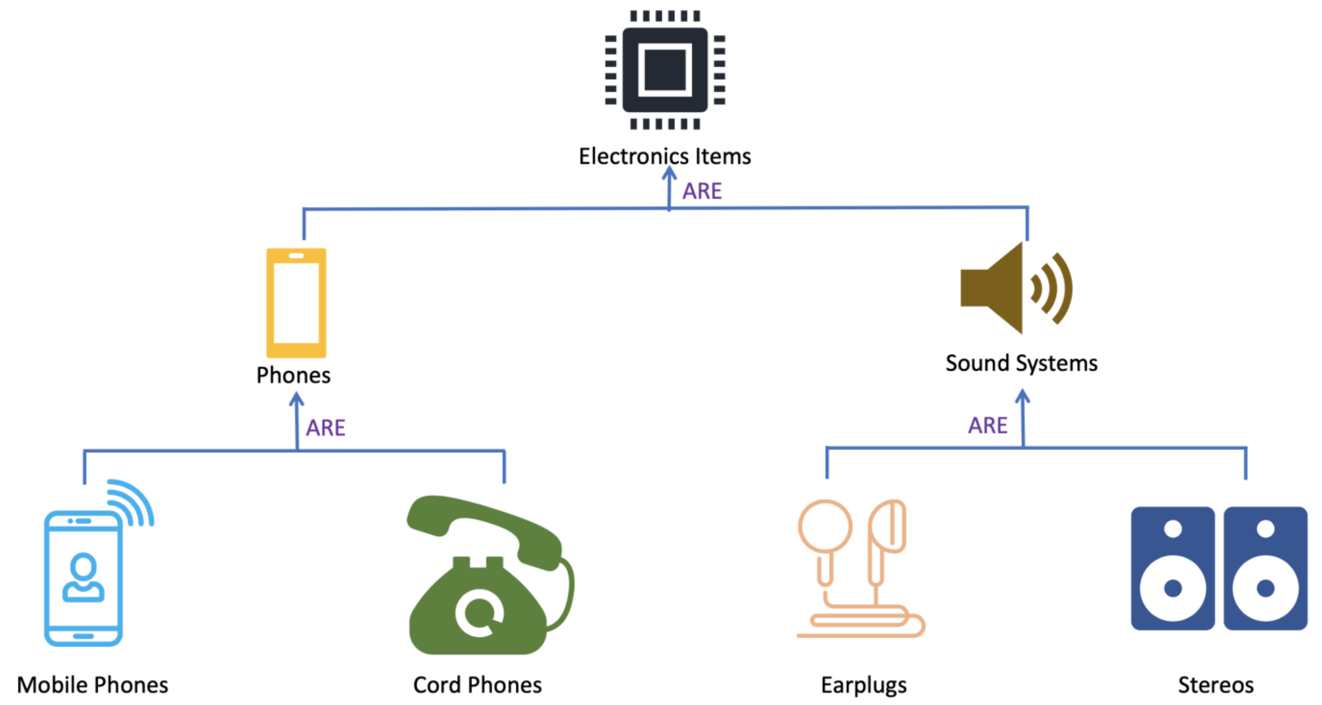
AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

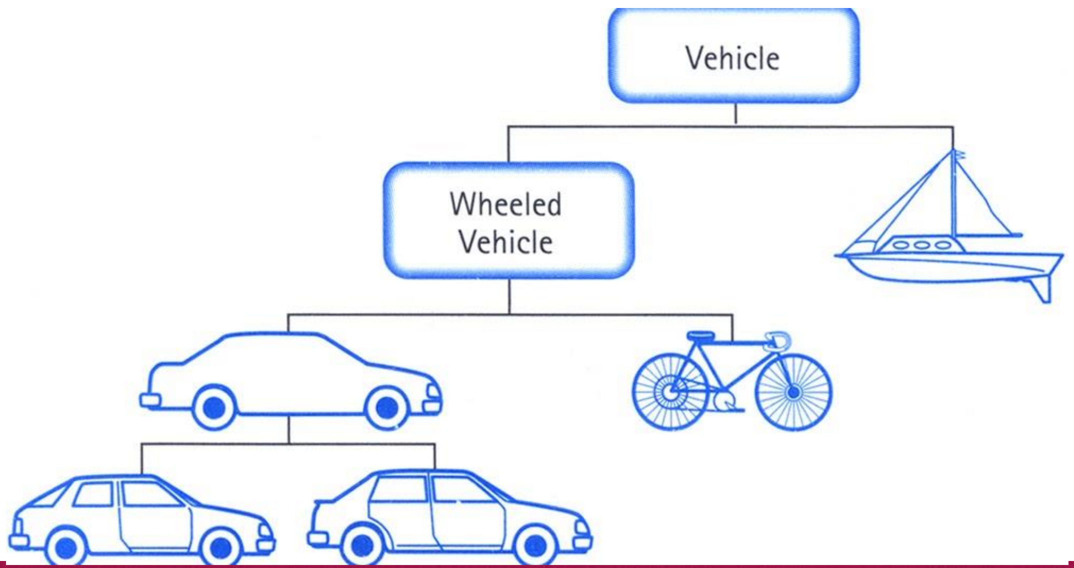
# 19CSE204
# Object Oriented Paradigm
# 2-0-3-3

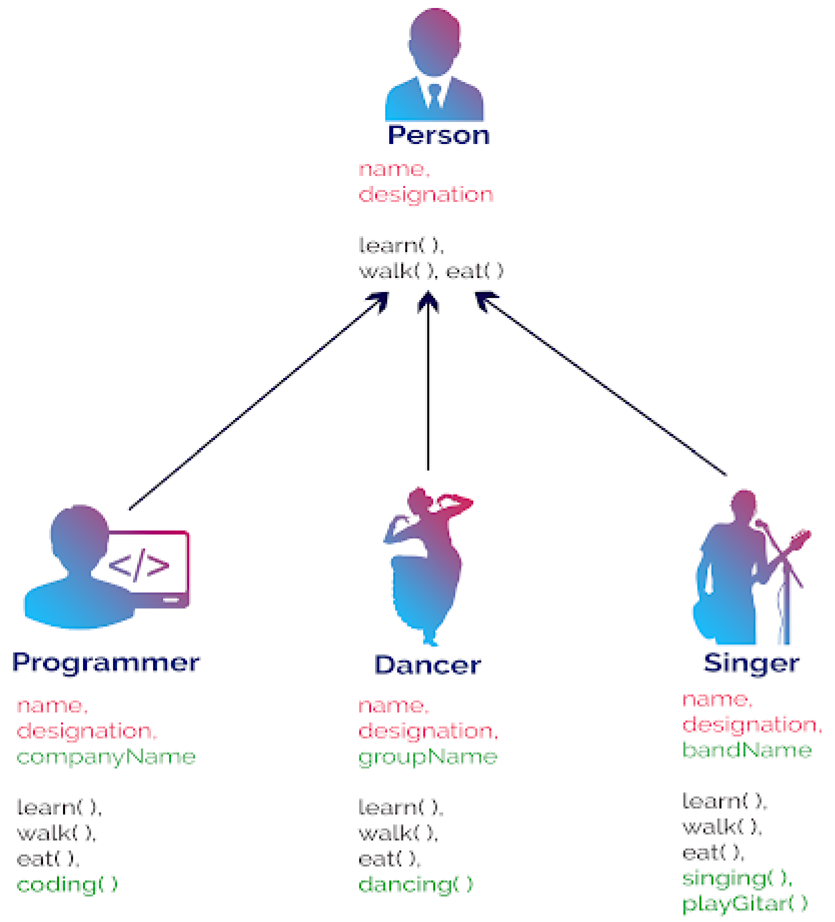
# Inheritance in Java

-the ability in **Java** for one class to **inherit** from another class.

# Real world examples- Inheritance







# Inheritance ( Another corner stone of OOPS)

- **Java inheritance** refers to the ability in **Java** for one class to **inherit** from another class.
- In **Java** this is also called extending a class. One class can extend another class and thereby **inherit** from that class.
- When one class **inherits** from another class in **Java**, the two classes take on certain roles.
    - **Sub Class/Child Class**: **Subclass** is a **class** which inherits the other **class.** It is also called a derived **class**, extended **class**, or child **class**.
- .
    - **Super Class/Parent Class**: **Superclass** is the **class** from whereas **subclass** inherits the features. It is also called a **base class** or a **parent class**.

Inheritance promotes code reusability

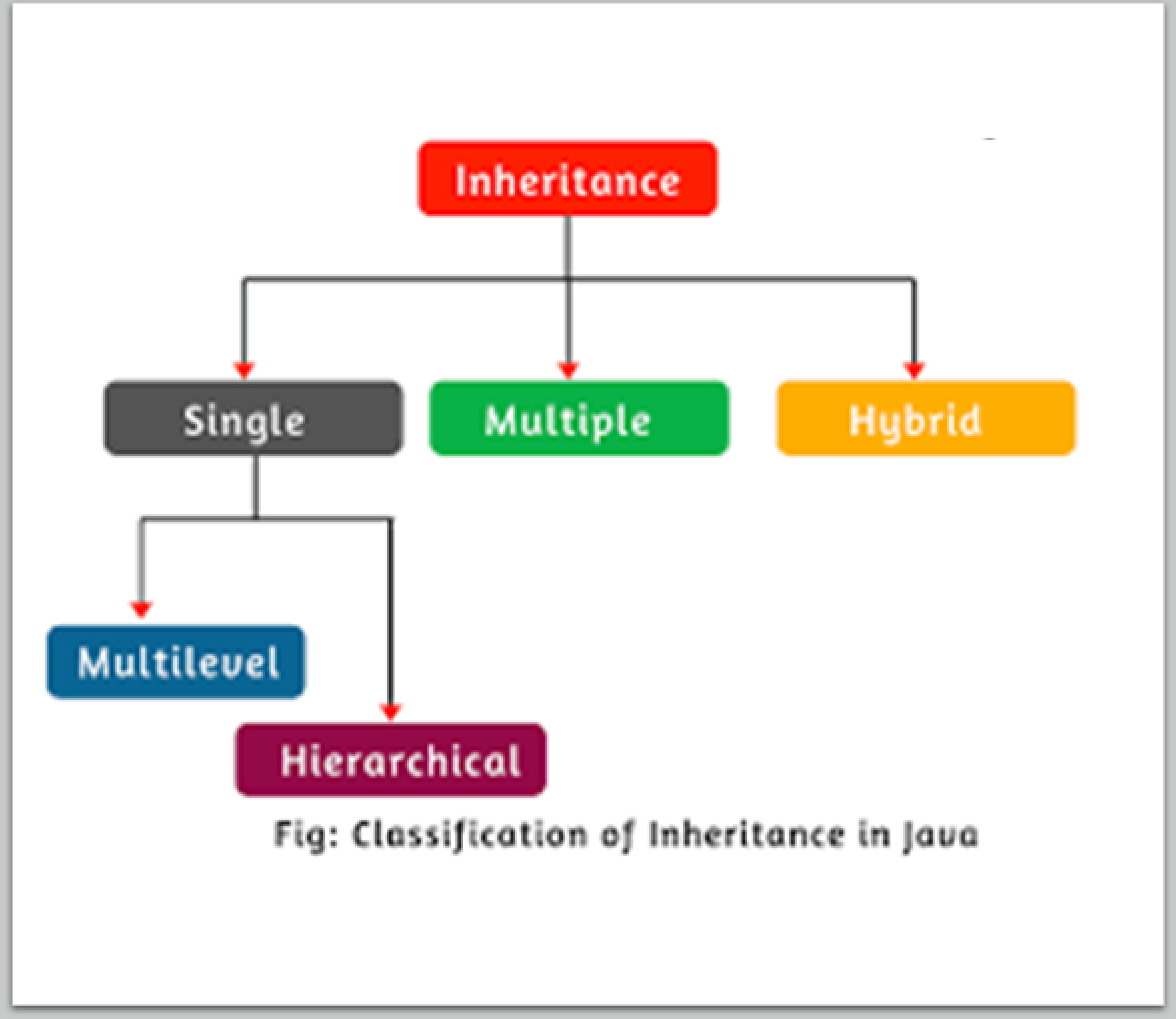


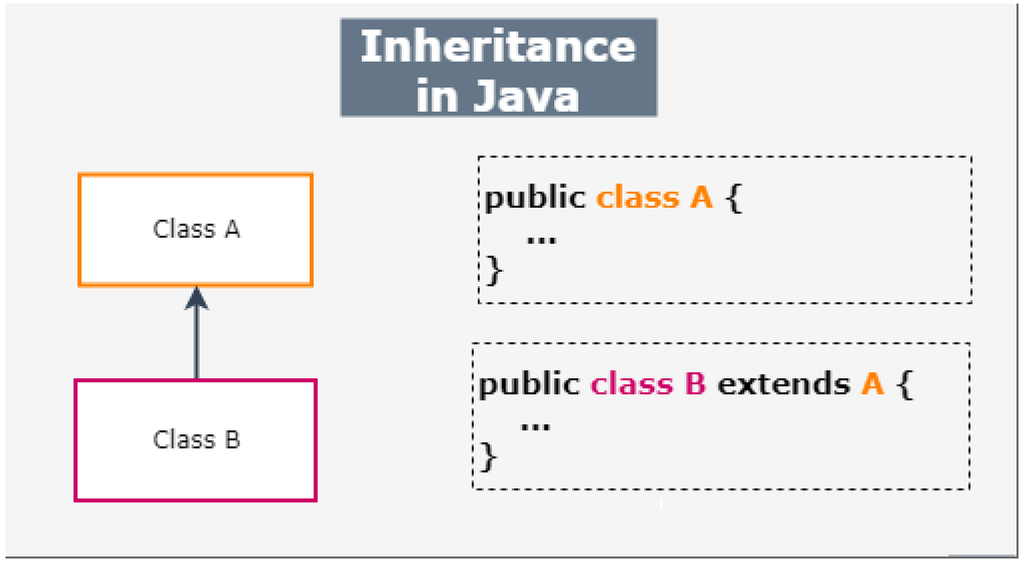
Fig: Classification of Inheritance in Java

## Types of Inheritance

A class which is inherited is called a **parent** or **superclass**, and the new class is called **child** or **subclass**.

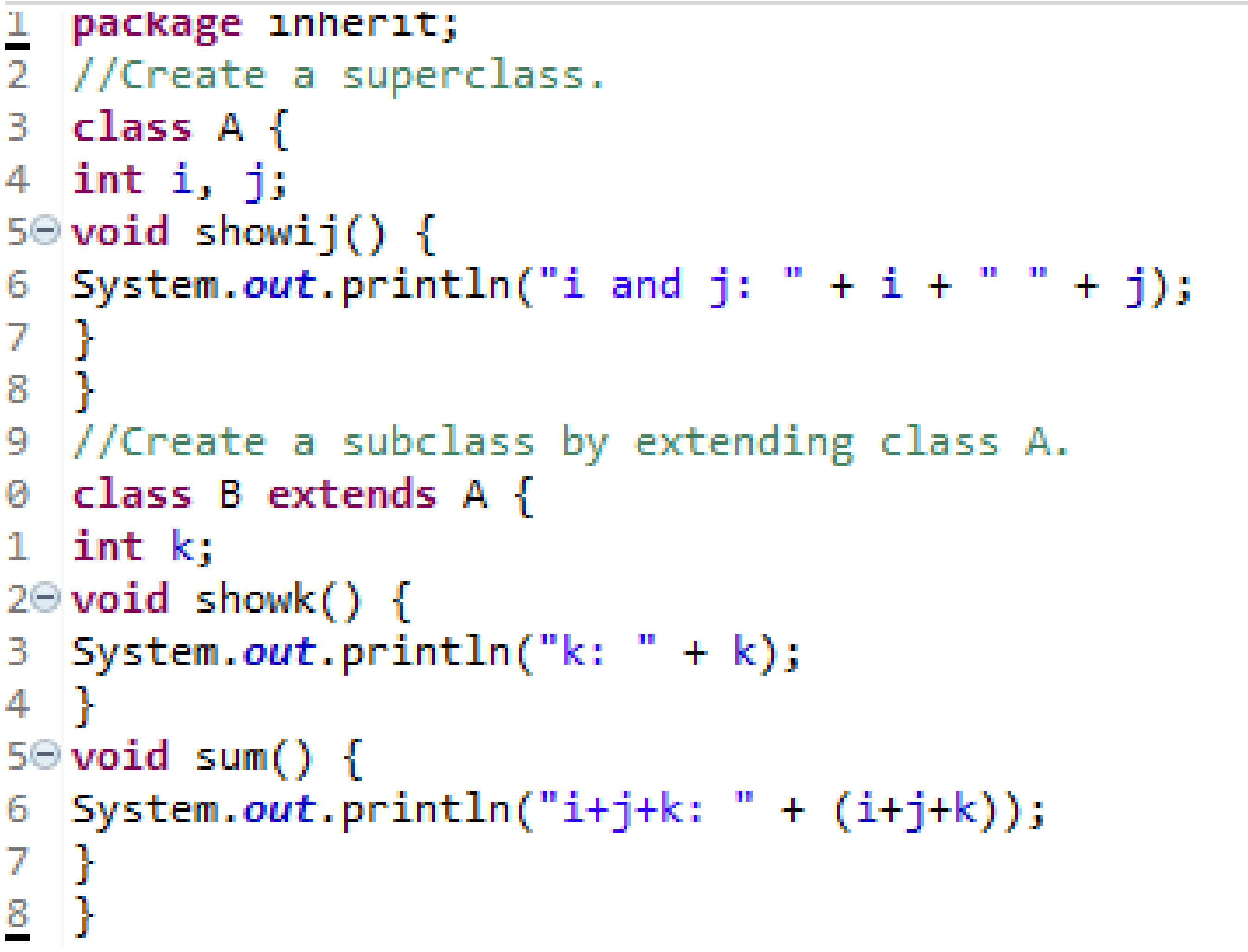
# A simple example of inheritance

class *subclass-name* extends *superclass-name* {
// body of class
}



```
public class A {
    ...
}
```

```
public class B extends A {
    ...
}
```



```
package inherit;
//Create a superclass.
class A {
int i, j;
void showij() {
System.out.println("i and j: " + i + " " + j);
}
}
//Create a subclass by extending class A.
class B extends A {
int k;
void showk() {
System.out.println("k: " + k);
}
void sum() {
System.out.println("i+j+k: " + (i+j+k));
}
}
```

```java
public class Driver {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        A superOb = new A();
        B subOb = new B();
        // The superclass may be used by itself.
        superOb.i = 10;
        superOb.j = 20;
        System.out.println("Contents of superOb: ");
        superOb.showij();
        System.out.println();
        /* The subclass has access to all public members of
        its superclass. */
        subOb.i = 7;
        subOb.j = 8;
        subOb.k = 9;
        System.out.println("Contents of subOb: ");
        subOb.showij();
        subOb.showk();
        System.out.println();
        System.out.println("Sum of i, j and k in subOb:");
        subOb.sum();
        }
```

**Output**

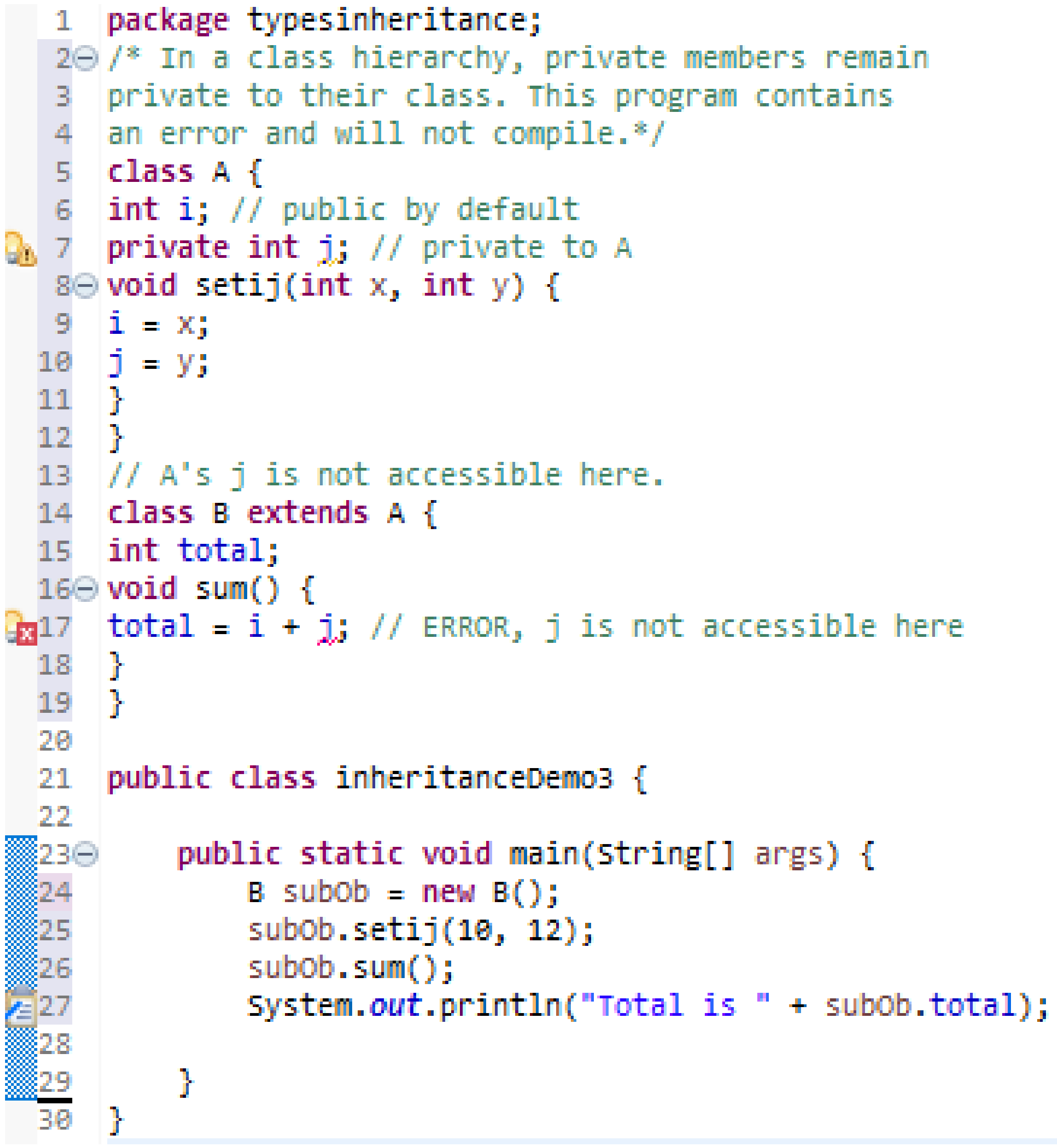
```
Contents of superOb:
i and j: 10 20

Contents of subOb:
i and j: 7 8
k: 9

Sum of i, j and k in subOb:
i+j+k: 24
```

# Member Access and Inheritance

- Although a subclass includes all of the members of its superclass, it cannot access those members of the superclass that have been declared as private.

Solution : Give the access modifier protected to j

This gives an error as j is not accessible in class B, as it is declared as private in A



```
package typesinheritance;
/* In a class hierarchy, private members remain
private to their class. This program contains
an error and will not compile.*/
class A {
int i; // public by default
private int j; // private to A
void setij(int x, int y) {
i = x;
j = y;
}
}
// A's j is not accessible here.
class B extends A {
int total;
void sum() {
total = i + j; // ERROR, j is not accessible here
}
}

public class inheritanceDemo3 {

    public static void main(String[] args) {
        B subOb = new B();
        subOb.setij(10, 12);
        subOb.sum();
        System.out.println("Total is " + subOb.total);

    }
}
```

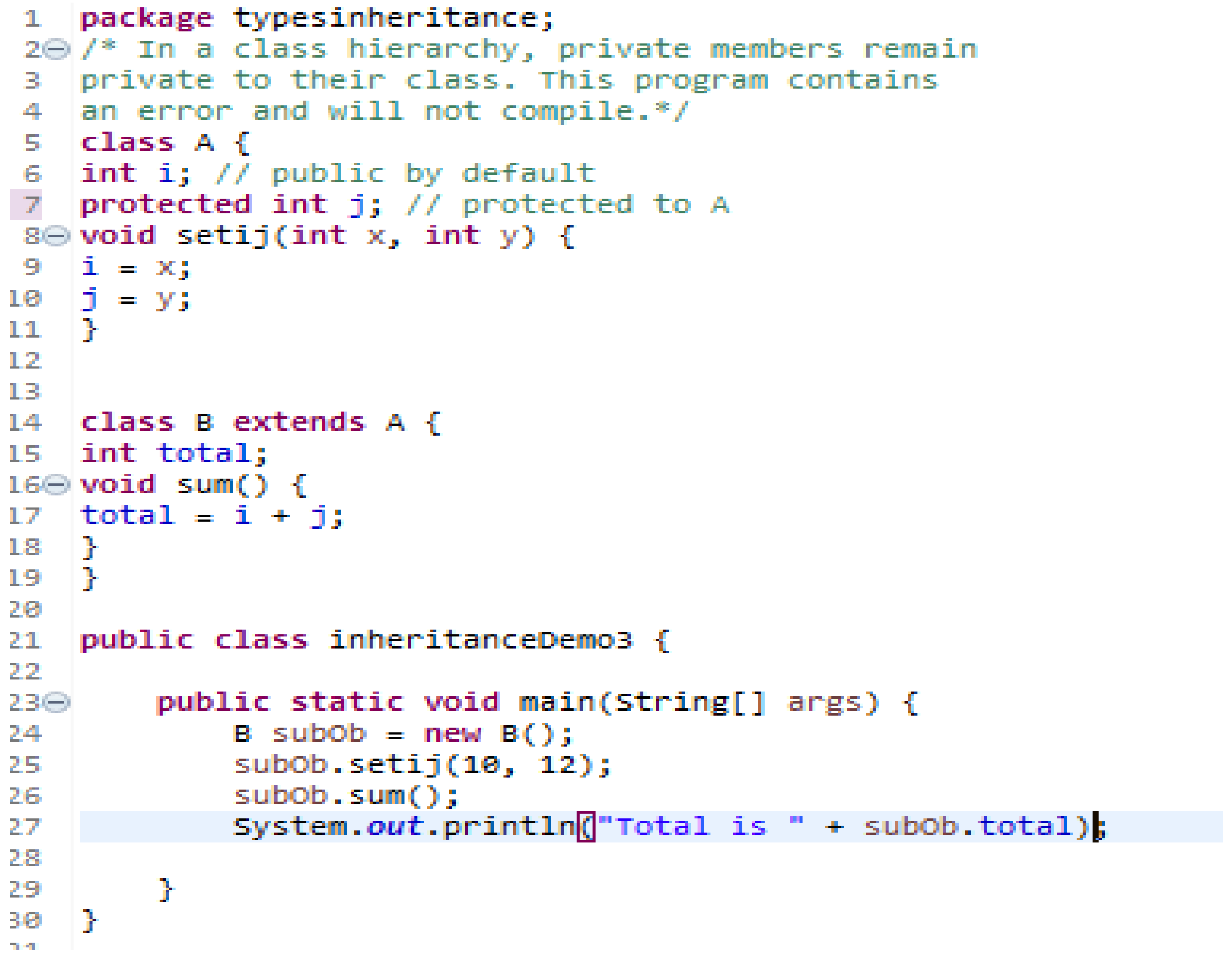
## Access specifiers

| | default | private | protected | public |
|---|---|---|---|---|
| Same Class | Yes | Yes | Yes | Yes |
| Same package subclass | Yes | No | Yes | Yes |
| Same package non-subclass | Yes | No | Yes | Yes |
| Different package subclass | No | No | Yes | Yes |
| Different package non-subclass | No | No | No | Yes |

**Protected:** The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.

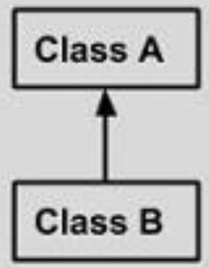
## Error rectified

- Modified program in slide 8
- Line 7- j is made protected, hence possible to access within the same package subclass



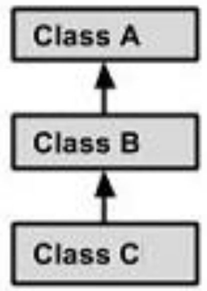
```
1  package typesinheritance;
2  /* In a class hierarchy, private members remain
3  private to their class. This program contains
4  an error and will not compile.*/
5  class A {
6  int i; // public by default
7  protected int j; // protected to A
8  void setij(int x, int y) {
9  i = x;
10 j = y;
11 }
12
13
14 class B extends A {
15 int total;
16 void sum() {
17 total = i + j;
18 }
19 }
20
21 public class inheritanceDemo3 {
22
23     public static void main(String[] args) {
24         B subOb = new B();
25         subOb.setij(10, 12);
26         subOb.sum();
27         System.out.println("Total is " + subOb.total);
28
29     }
30 }
```

**Single Inheritance**



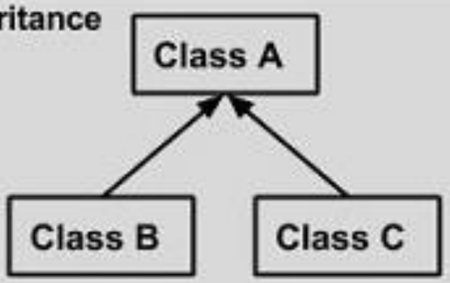
```
public class A {
   ........
}
public class B extends A {
   .........
}
```

**Multi Level Inheritance**



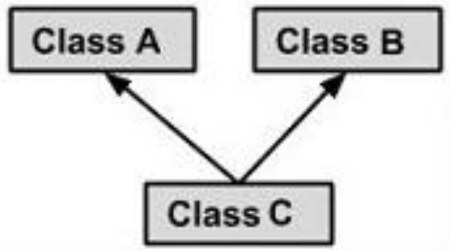
```
public class A { ..................}

public class B extends A {...................}

public class C extends  B {..................... }
```

**Hierarchical Inheritance**



```
public class A { ..................}

public class B extends A {...................}

public class C extends A {..................... }
```

**Multiple Inheritance**



```
public class A { ..................}

public class B {...................}

public class C extends  A,B {
   ......................
} // Java does not support mutiple Inheritance
```

Namah Shivaya!