iamneo

Inheritance and Encapsulation

www.iamneo.ai

Buildings blocks of JAVA

- Inheritance
- Encapsulation
- Abstraction
- Polymorphism
- Class and object

Need for Inheritance!

```
class A
{
     void m1()
     {
     }
     void m2()
     {
     }
}
```

m1 , m2 defined twice

```
class B
{
     void m1()
     {
     }
     void m2()
     {
     }
     void m3()
     {
     }
     void m4()
     {
     }
}
```

Rewriting Code:

Duplication of the same code, which is not allowed in real-time applications → The length of the code is increased

Therefore we go for Inheritance

Same code with Inheritance:

```
class A
{
    void m1()
    {
    }
    void m2()
    {
    }
}
class B extends A
{
    void m3()
    {
    }
    void m4()
    {
    }
}
```

Parent class
or
Super-class
or
base class

Child class
or
Sub-class
or
derived class

Purpose of extends!

- To provide a relationship between 2 classes
- To implement reusability and reduce redundancy

Is Multiple Inheritance supported in JAVA?

```
class A
{
    void m1()
    {
    }
    void m2()
    {
    }
}
class B
{
    void m3()
    {
    }
    void m4()
    {
    }
}
```

```
class C
{
    void m5()
    {
    }
    void m6()
    {
    }
}
```

What if class C tries to inherit the property of classes A and B?

```
class B extends A, B
```

ERROR!

Multiple Inheritance is not supported in JAVA

Inheritance objects and methods:

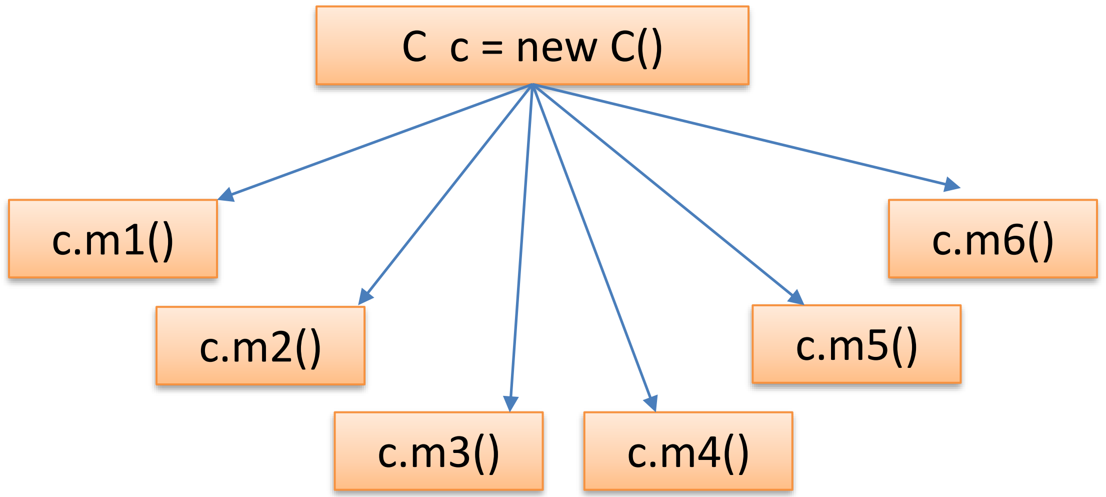
```
class A
{
    void m1()
    {
    }
    void m2()
    {
    }
}
class B extends A
{
    void m3()
    {
    }
    void m4()
    {
    } }
```

→ Can access 2 methods

→ Can access 4 methods

```
class C extends B
{
    void m5()
    {
    }
    void m6()
    {
    }
```

Creating object of class C



Can access all 6 methods

Input Stream Hierarchy:

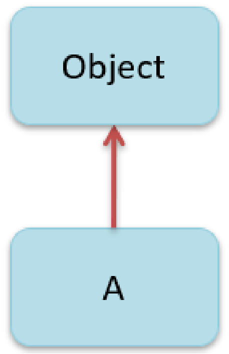
At the project level, for which class, the object should be created?

Child class

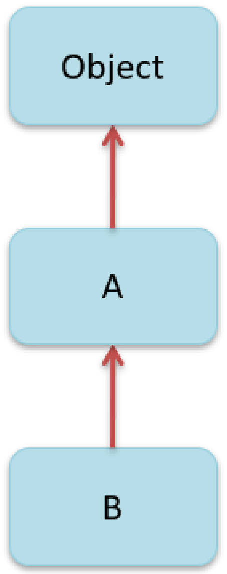
One can access Parent and child properties using the object of the child class!

Representation of Inheritance:

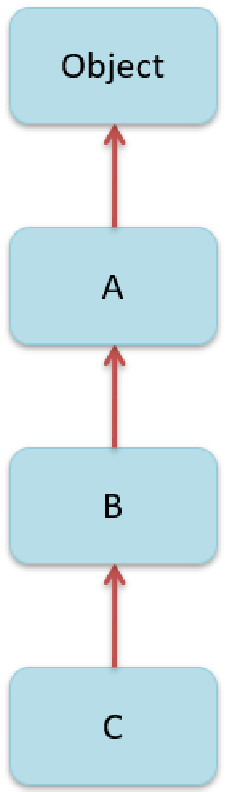
Directly inherited from the Object class



Indirectly inherited from the Object class

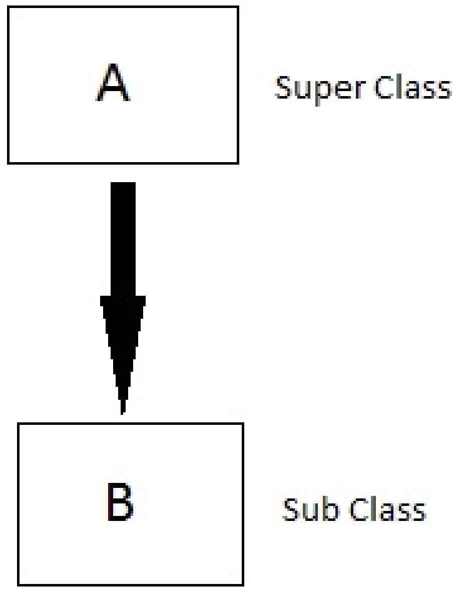


Indirectly inherited from the Object class



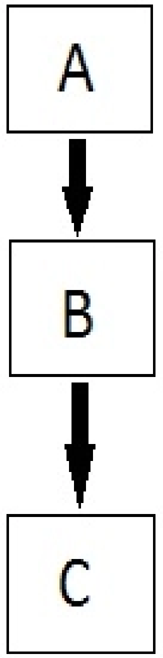
Inheritance Types:

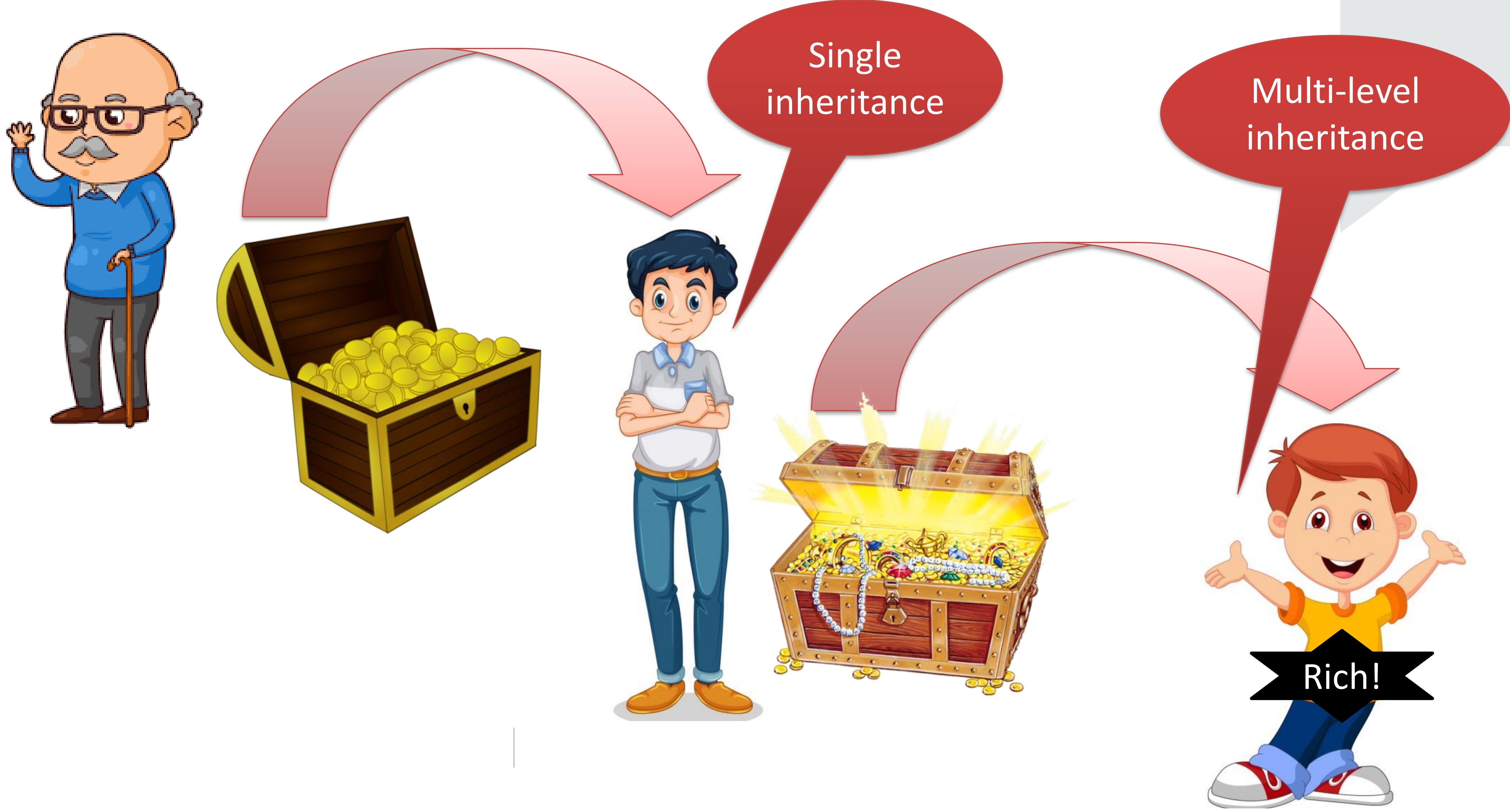
1) Single Inheritance



Inheritance Types:

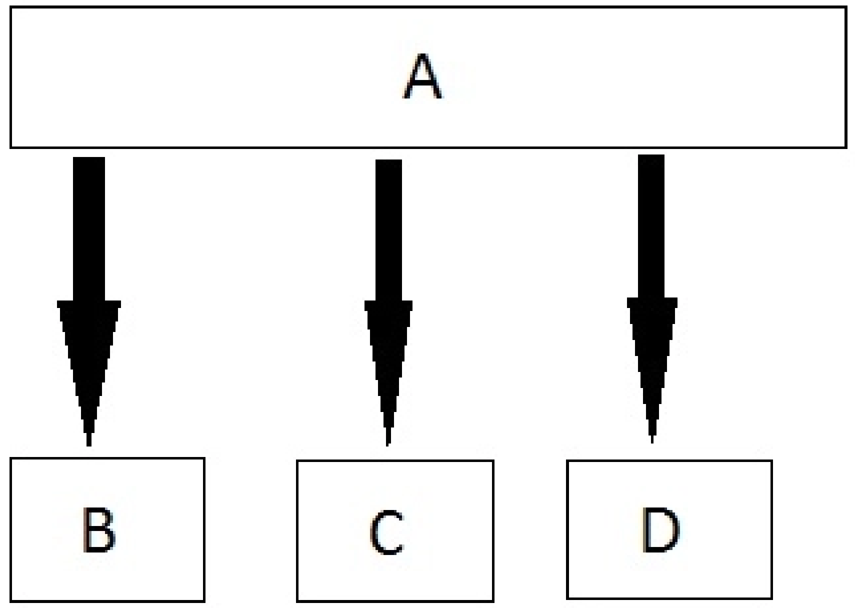
2) Multilevel Inheritance


$$\begin{tikzcd}\boxed{A} \arrow[d] \\ \boxed{B} \arrow[d] \\ \boxed{C}\end{tikzcd}$$


Single inheritance
Multi-level inheritance
Rich!

Inheritance Types:

3) Hierarchical Inheritance



Encapsulation:

- Encapsulation in Java is the process by which data (variables) and the code that acts upon them (methods) are integrated as a single unit.
- By encapsulating a class's variables, other classes cannot access them, and only the methods of the class can access them.
- It provides you the **control over the data**.
- It is a way to achieve **data hiding** in Java because other classes will not be able to access the data through the private data members.

Encapsulation:

```
class Car
{
    int mileage;
    int max_speed;
    public:
        void display();
}
```

Data Encapsulation - Wrapping the data and functions in one single unit

JAVA Package:

- A package is a group of similar types of classes, interfaces, and sub-packages
- Contains built-in packages and user-defined package and built-in packages such as java, lang, awt, javax, swing, net, io, util, SQL, etc. (14 packages)

```
package mypack;

public class Simple
{
            public static void main(String args[])

            {  System.out.println("Welcome to package");     }
}
```

Access Package from other package

Three ways to access the package from outside the package:

1. import package.*;
2. import package.classname;
3. Fully qualified name.