INHERITANCE

Recap 

Q1. What is Boxing in java

Placing a primitive variable in an object is known as boxing and This allows the primitive to be used where objects are required.

Recap

How to convert String to Integer type

- ✓ Integer.valueOf(string) : Integer
- ✓ Integer.parseInt(string) : int

OBJECTIVES

On completion of this topic, you will be able to:

- Define Inheritance
- Implement Inheritance in Java
- Use the Access Specifier Protected
- List the type of Inheritance
- Implement method overriding concept in Inheritance

INHERITANCE

- Inheritance is one of the major features of OO Concept.
- Inheritance is the process in which one object can acquire the properties and behaviour of the parent object.
- Using inheritance new classes can be built on already existing classes, so that the new class can reuse methods and fields present in the parent class.
- Inheritance represents an IS-A relationship, also known as parent-child relationship.

INHERITANCE

Example:

	Mountain Bike	Road Bike	Tandem Bike
Common Features : Speed, Pedal Cadence, Gear.			
Special Features:	Additional chain ring, Thick tyres	Drop handlebars, Shock absorbers	Two seats and two handlebars

INHERITANCE

Advantages :

Code Reusability,
Easy Maintenance.

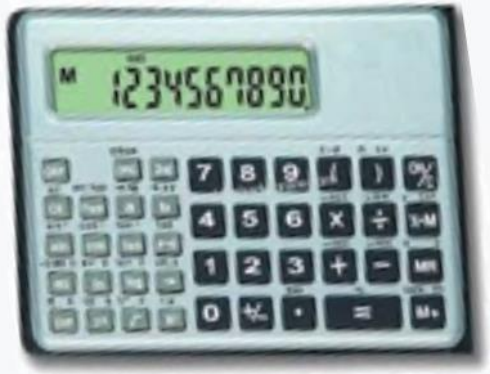
Notation for Inheritance

SimpleCalculator

Add
Subtract
Multiply
Divide

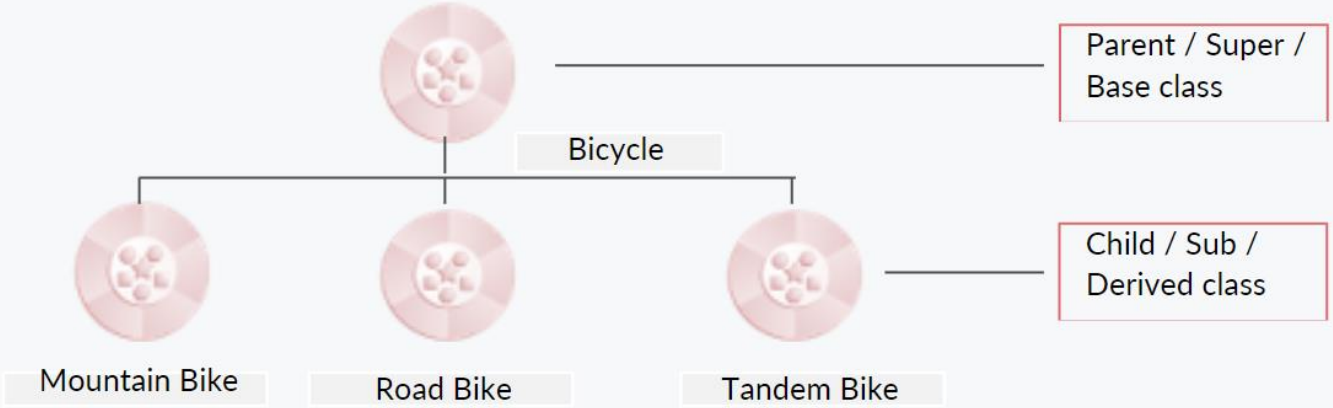
ScientificCalculator

calculateSine
calculateCos
calculateLog
calculateSqrt



INHERITANCE

In this example, take the common features and behavior of the objects and frame a new class, say, Bicycle.



By Inheritance, the members of parent class can be **reused** by all the sub classes. The sub class can have its own attributes and methods specific to them.

By Inheritance, any changes in a method, can be done in parent class. That change gets reflected to all the child classes. This makes **maintenance easy**.

INHERITANCE

Example: Without Inheritance

```java
public class Employee {


    private int employeeId;
    private String name;

    public Employee()
    {
       System.out.println("In Employee "
            + "Constructor");
    }
    //Assume Getters and setters for all attributes

}
```

```java
public class PermanentEmployee {


    private int employeeId;
    private String name;
    private int basicPay;


    public PermanentEmployee()
    {
       System.out.println("In PermanentEmployee "
            + "Constructor");
    }
    //Assume Getters and setters for all attributes

}
```

- PermanentEmployee class has all attributes and methods (getter and setter) that are already available in Employee class.

  Results in duplication of code.
- Any changes to the common methods has to be done in both the classes.

INHERITANCE

Example: With Inheritance

- Permanent Employee is a type of employee. So there is an Inheritance Relationship.
- Use "extends" keyword to make a class inherit another class.

```
public class PermanentEmployee extends Employee
{
    private int basicPay;

    public PermanentEmployee()
    {
       System.out.println("In PermanentEmployee "
            + "Constructor");
    }
    //Assume Getters and setters for basicPay

}
```

- Duplication of code is avoided
- Child class inherits the members of parent class.
- Child class can have additional attributes and methods specific to that class.

INHERITANCE

Example: With Inheritance

```
public class EmployeeUtility {

    public static void main(String a[]) {

        PermanentEmployee per_empObj=new PermanentEmployee();

    }
}
```

- On creating an object for child class,
  - the parent class constructor is executed first
  - then the child class constructor gets executed

- *A class can inherit only one class.*

Output :

In Employee Constructor
In PermanentEmployee Constructor.

USAGE OF PROTECTED

How will you access private members of parent class in a child class?

Use the public methods to access them.

Example code :

To access employeeId and name, use the public getters.

If the access specifier for employeeId and name is default, they can be accessed directly as employeeId and name, provided the sub class is in the same package.

```
public class PermanentEmployee extends Employee
{
    private int basicPay;

    //Assume Getters and setters for basicPay

    public PermanentEmployee()
    {
        System.out.println("In PermanentEmployee "
                + "Constructor");
    }

    public void display() {

        System.out.println("ID "+getEmployeeId()+
                "  Name "+getName()+"  Basic Pay  "+basicPay);

        //As fields are private use public methods to access
    }
}
```

USAGE OF PROTECTED

Is it possible to access the members of parent class directly, in child class?

Yes, by making the access specifier as protected.

Protected members can be accessed by all classes within the same package where it is declared and also by subclasses in other packages.

Protected access specifier can be applied for attributes, methods and constructor. They cannot be applied for class and interface.

ACCESS CONTROL

Access control based on modifiers on class member declarations are listed here:

MODIFIER	ACCESS LEVELS			
	Same Class	Same Package	Subclass	Universe
Private	**Yes**	No	No	No
Default	**Yes**	**Yes**	No	No
Protected	**Yes**	**Yes**	**Yes**	No
Public	**Yes**	**Yes**	**Yes**	**Yes**

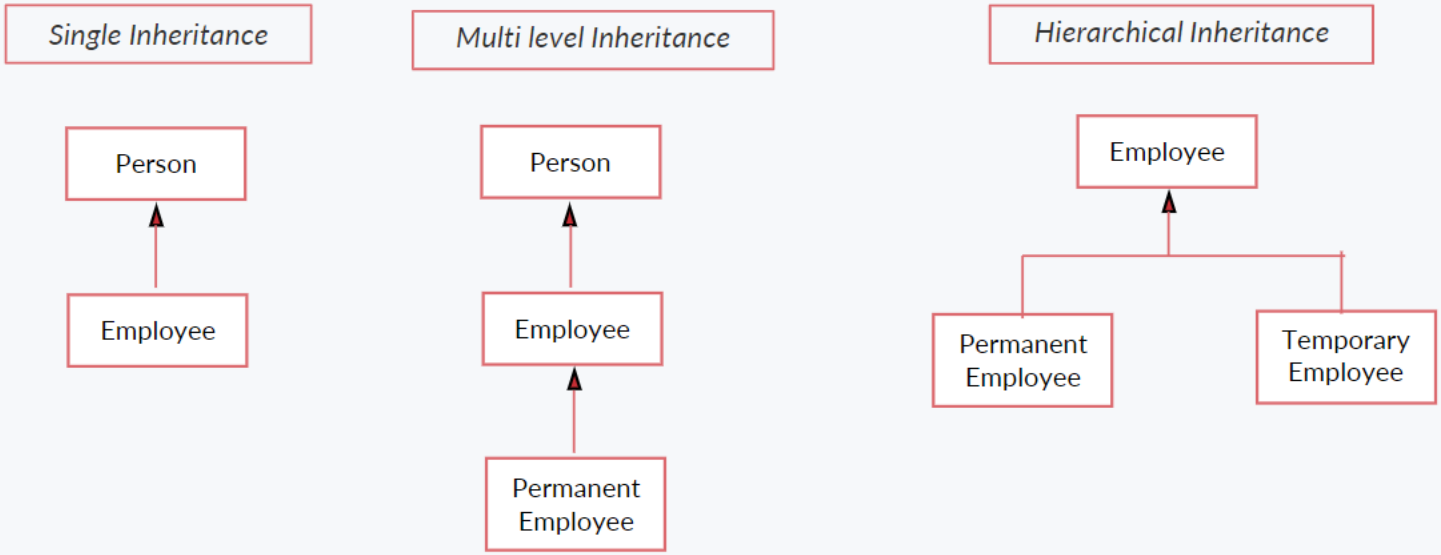
TYPES OF INHERITANCE

TYPES OF INHERITANCE

Single Inheritance - A class inherits only one class.

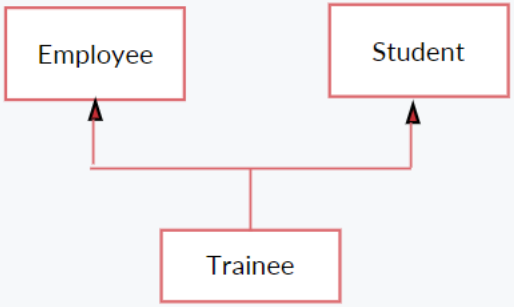
Multi level Inheritance - The child of one class is a parent of another class.

Hierarchical Inheritance - A class is a parent of more than one class.



TYPES OF INHERITANCE

Multiple Inheritance - A class inherits more than one class.



MULTIPLE INHERITANCE NOT SUPPORTED IN JAVA

METHOD OVERRIDING

Example Scenario:

Daughter inherits all the properties of Mom.

Mom is the super class and Daughter is the sub class.

Assume

Mom likes to hear music. Daughter also likes music – (inherits).

What if Mom likes melody, whereas Daughter likes to listen to Pop

Daughter overrides the behavior of Mom.

METHOD OVERRIDING

- A subclass can have a method with same name, argument list and return type (except co-variant type) as declared in the parent class, but with a different implementation.
- When the overridden method is invoked using a sub class object, it will override the method in the parent class and execute the method in the child class. ***Hence the concept Method overriding.***
- Rules for Method Overriding:
  - Method name should be same as in the parent class.
  - Methods argument list should be the same as in parent class.
  - Same return type (or its sub type).
  - The subclass overridden method cannot have weaker access than super class method.
    Example: If parent class method is declared as public, then the overridden method in sub class cannot be private or protected or default.

HANDS ON TRAINING

Doubts Time

QUIZ TIME

Quiz Time

Q1. Which Inheritance type not supported in java

A Single Inheritance

B Hierarchy Inheritance

C Multiple Inheritance

D Multi-Level Inheritance

Quiz Time

Which Keyword is used to call the base class constructor from child class constructor

A base

B this

C super

D extends

SUMMARY

SUMMARY

Having completed this module, you should be able to:

- Define Inheritance
- Implement Inheritance in Java
- Use the Access Specifier Protected
- List the type of Inheritance
- Implement method overriding concept in Inheritance