

Inheritance

Object Oriented Programming

by Dr. Rajendra Prasath

Indian Institute of Information Technology
Str City – 517 646, Andhra Pradesh, India

Recap: Objects in JAVA?

- An entity that has state and behaviour is known as an object
 - ♦ Examples: Chair, bike, marker, pen, table, car etc.
 - ♦ It can be physical or logical
- ♦ An object has three characteristics:
 - ♦ State: represents data (value) of an object
 - ♦ Behaviour: represents the behaviour (functionality) of an object such as deposit, withdraw and so on
 - ♦ Identity (Internally used):
 - ♦ Signature (unique) of the object
 - ♦ Object identity is typically implemented via a unique ID
 - The value of the ID is not visible to the external user
 - ♦ But, Internally by JVM to identify each object uniquely



22/08/19

Recap: Nested Classes

- Member Inner Class: A class created within class and outside method
- ♦ Local Inner Class: A class created within method
- Anonymous Inner Class: A class created for implementing interface or extending class
 - ♦ Its name is decided by the java compiler
- Static Nested Class: A static class created within class



Recap: Static Nested Class

```
class Outer {
       static int id=11;
       static String name="";
       static class Inner {
              void display() {
                     System.out.println("ID is: "+id);
                     System.out.println("Name is: "+name);
       public static void main(String args[] {
              Outer.Inner obj=new Outer.Inner();
              obj.dispaly();
```

Recap: Member Inner Class

```
class Outer {
       private static int age=28;
       class Inner {
               int getAge() {
                       return age;
       void displayAge() {
               Inner in = new Inner();
               System.out.println("Age = " + in.getAge() );
       public static void main(String args[] {
               Outer out = new Outer();
               out.displayAge();
```

Recap: Local Inner Class

```
class Outer{
       private int data=30; //instance variable
       void display(){
               int value=50; //local variable
               class Local {
                       void msg(){
                               System.out.println("Value is: " + value);
               Local I = new Local();
               1.msg();
       public static void main(String args[]){
               Outer obj = new Outer();
               obj.display();
```

22/08/19

Recap: Anonymous Inner Class

```
Interface Message{
      String welcome();
public class CodeTester {
       public void showMessage(Message m) {
              System.out.println(m.welcome());
       Public static void main(String args[]) {
              CodeTester ct = new CodeTester();
              ct.showMessage(new Message() {
                     public String welcome() {
                             return "Welcome Folks!!";
              });
```



Inheritance in JAVA

In this class, we will look into this aspect of Object Oriented Programming, specifically the approaches in JAVA





Inheritance in JAVA

- ♦ Inheritance in java is a mechanism in which an object acquires all the properties and behaviors of the parent object
- When a Class extends another class it inherits all non-private members including fields and methods.
- ♦ Inheritance in Java can be best understood in terms of Parent and Child relationship
 - ♦ Super class(Parent) and
 - ♦ Sub class(Child)
- ♦ Inheritance defines is-a relationship between a Super class and its Sub class.
- ♦ extends and implements keywords are used to describe inheritance in Java



22/08/19

IS-A relationship

- ♦ Known as Parent Child Relationship
- ♦ Why do we need Inheritance?
 - ♦ For Method Overriding
 - Enable polymorphism to reuse code for different classes by putting it in a common super class
 - ♦ Efficient code Re-usability
- Super Class providing non-private data members and methods - known as base or super or parent class
- Sub Class: The class that takes up the data members and methods from Super class is known as sub or derived or child class
- The data members and methods of a class are known as features
- The concept of inheritance is also known as extendable classes or sub classes





Inheritance - By Definition

♦ Inheritance:

Inheritance is a process where one object acquires the properties of another object

♦ Super class:

The Class whose properties are not derived from any other class

♦ Sub class:

The Class which inherits the properties of another object is called a **Subclass**

♦ Keywords:

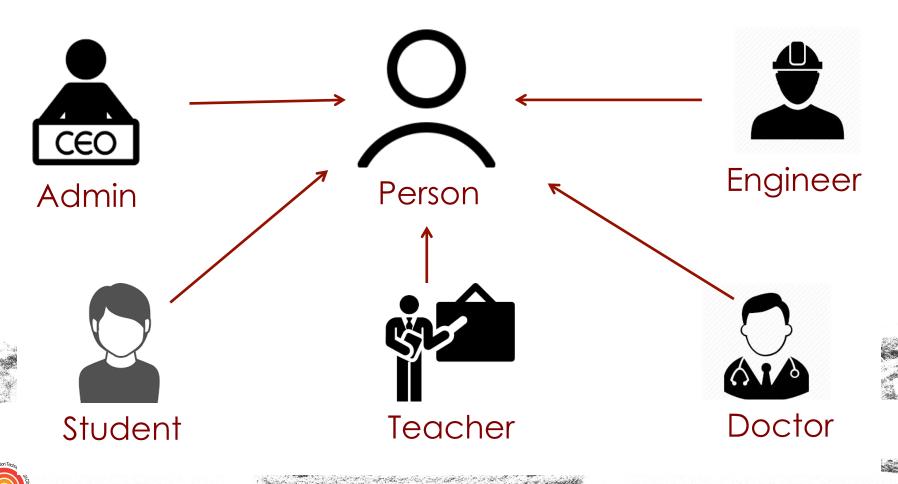
extends and implements





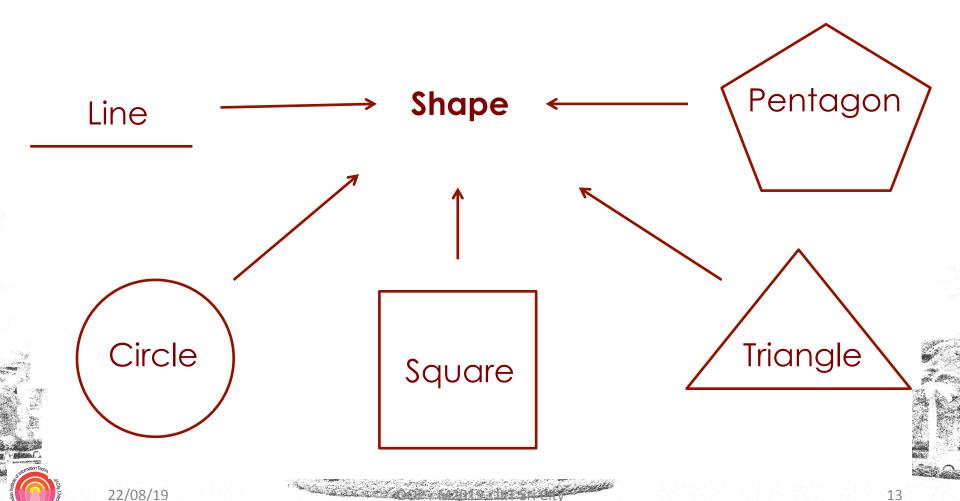
Inheritance – Aan Example

♦ Let us look at an example:



Inheritance – Another Example

♦ Let us look at another example:



Exercise - Class Room

♦ Two Exercises ():

- ♦ A) Can you create class diagrams for the first example?
 - ♦ Person
 - ♦ Student
 - ♦ Teacher
 - ♦ Doctor
 - ♦ Engineer
 - ♦ Administrator
- ♦ B) Create Class Diagram for Geometric Objects
 - ♦ Shape
 - ♦ Line
 - ♦ Circle
 - ♦ Shape
 - ♦ Triangle
 - Pentagon and so on





Reusability

- ♦ REUSE is one of the main purposes of Inheritance
- One can easily ass new classes by inheriting the properties from the existing class
 - Select an existing class closer to the desired functionality
 - Create a new class and inherit it from the selected class
 - ♦ Add or Modify the inherited functionality



How to do?

- Which properties would be a part of the parent class?
 - Common properties (or essential properties in specific sense)

- Which properties would be a part of the child class?
 - ♦ Specific to the inherited class
 - ♦ Specific functionalities



Core Concepts with Inheritance

Let us look at the following concepts:

♦ Generalization

♦ Subtyping (extension)

♦ Specialization (restriction)



Generalization

In OO models, some classes may have common characteristics

We extract these features into a new class and inherit original classes from this new class

This concept is known as Generalization



Generalization – An Example

♦ Let us look at the Class Diagrams

Line

Color

Vertices

Length

Move()
setColor()
getLength()

Circle

Color

Vertices

Radius

Move()
setColor()
computeAea()

Triangle

Color

Vertices

angle

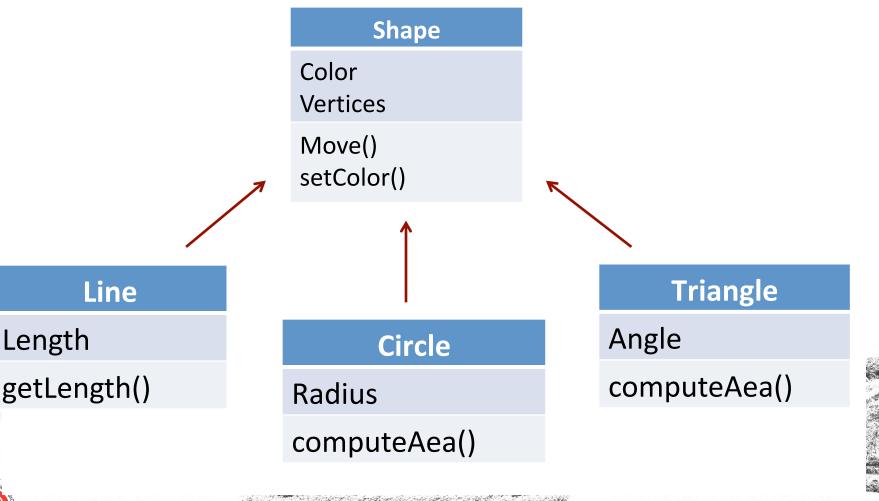
Move()
setColor()
computeAea()





Generalization – An Example

♦ Create a Class and inherit its properties





Length

Sub-typing and Specialization

- How to do Sub-typing & Specialization?
 - ♦ We want to add a new class to an existing model
 - Find an existing class that already implements some of the desired states and behaviors
 - ♦ Inherit a new class from this class
 - ♦ add unique behavior to the new class
- Sub-typing means that derived class is behaviourally compatible with the base class



Sub-Typing – An Example

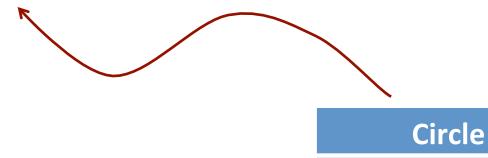
Shape

Color

Vertices

Move()

setColor()



Radius

computeAea()



Sub-typing and Specialization

♦ Sub-typing (Extension)

- Sub-typing means that derived class is behaviourally compatible with the base class
- Behaviourally compatible means that base class can be replaced by the derived class

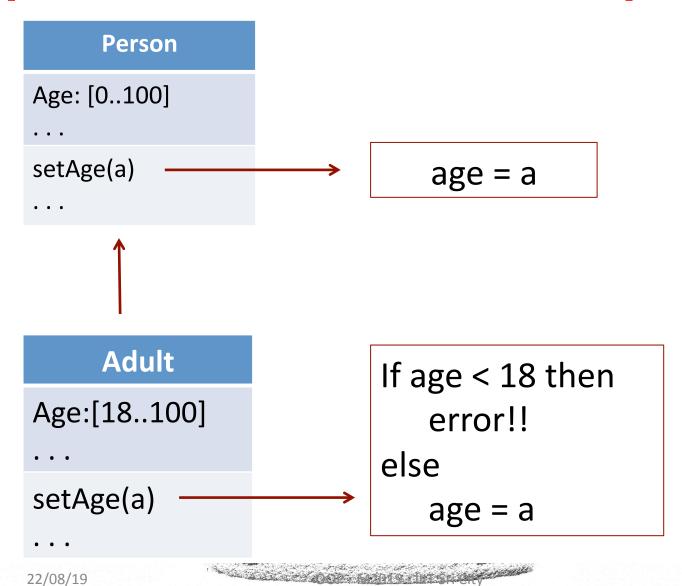
♦ Specialization (Restriction)

- Specialization means that derived class is behaviourally incompatible with the base class
- Behaviourally incompatible means that base class can't always be replaced by the derived class





Specialization – An Example





- ♦ Inheritance in Java is done using
 - ♦ extends In case of Java class and abstract class
 - → implements In case of Java interface.
- ♦ What is inherited?
 - In Java when a class is extended, sub-class inherits all the public, protected and default (Only if the sub-class is located in the same package as the super class) methods and fields of the super class.
- ♦ What is not inherited
 - Private fields and methods of the super class are not inherited by the sub-class and can't be accessed directly by the subclass.
 - Constructors of the super-class are not inherited. There is a concept of constructor chaining in Java which determines in what order constructors are called in case of inheritance.





♦ What is inherited?

In Java when a class is extended, sub-class inherits all the public, protected and default (Only if the subclass is located in the same package as the super class) methods and fields of the super class.

♦ What is not inherited?

- private fields and methods of the super class are not inherited by the sub-class and can't be accessed directly by the subclass.
- Constructors of the super-class are not inherited. There is a concept of constructor chaining in Java which determines the order in which constructors are called in case of inheritance.



- ♦ extends keyword
 - ♦ This keyword is used to inherit the properties of a class.

Following is the syntax of extends keyword:



- ♦ extends keyword
 - ♦ This keyword is used to inherit the properties of a class.

Following is the syntax of extends keyword:



IS-A Relationship - An example

♦ IS-A is a way of saying: This object is a type of that object

♦ Let us look at the following example:

public class Vehicle {
 Subclass
 public class TwoWheeler extends Vehicle {
 Subclass
 public class FourWheeler extends Vehicle {

IS-A relationship

TwoWheeler IS-A Vehicle FourWheeler IS-A Vehicle

Car IS-A FourWheeler

So

Car is a Vehicle

Subclass of both FourWheeler and Vehicle classes

public class Car extends FourWheeler {



Inheritance – An Example

♦ Let us look at the following example: Class Parent { public void p1() { System.out.println("Parent Method!!"); Class Child extends Parent{ public void c1() { System.out.println("Child Method!"); public static void main(String[] args) { Child ch = new Child(); ch.c1(); ch.p1();

Access Control and Inheritance

- ♦ A derived class can access all the non-private members of its base class.
- ♦ Base-class members that should not be accessible to the members of derived classes should be declared private in the base class.

The following rules for inherited methods are enforced:

- Methods declared **public** in a superclass also must be public in all subclasses
- Methods declared protected in a superclass must either be protected or public in subclasses; they cannot be private
- Methods declared private are not inherited at all, so there is no rule for them



Types of Inheritance – 5 types

Single inheritance **Class A** Class B → Multiple inheritance Class A Class B Class C Hierarchical inheritance Class B Class A Class C Multilevel inheritance Class A Class C **Class B** ♦ Hybrid inheritance **Class B Class D Class A** Class C

Single Inheritance - example

```
Class A {
       int data = 10:
Class B extends A {
       public void display() {
              System.out.println("Data is:" + data);
       public static void main(String[] args) {
              Bb = new B();
              b.display();
```

Multiple Inheritance - example

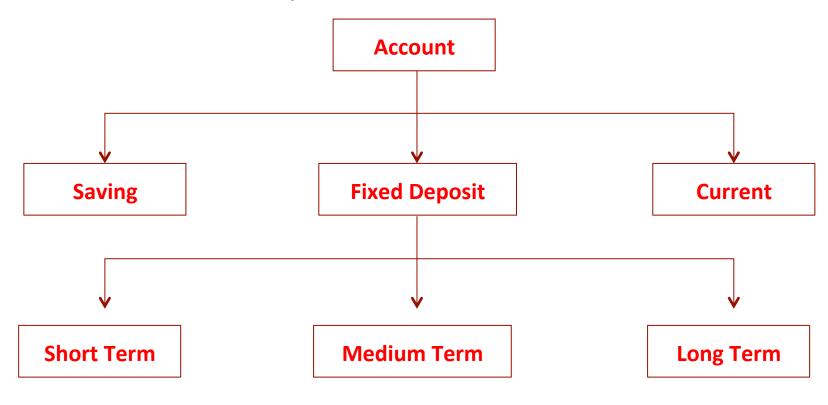
```
Class A {
         int data = 10:
Class B extends A {
         float salary = 30000.;
Class C extends B {
         public void display() {
                  System.out.println("Data is = " + data);
                  System.out.println("Salary is = " + salary);
         public static void main(String[] args) {
                  Cc = new C();
                  c.display();
```

Multiple Inheritance-Employee

```
Class Employee {
        float total = 0.0, salary = 10000.;
Class HRA extends Employee {
        float hra = 3000.:
Class DA extends HRA {
         float da = 5000.:
Class Income extends DA {
         float bonus = 3000.:
         public static void main(String[] args) {
                  Income net = new Income();
                  net.total = net.salary + net.hra + net.da + net.bonus;
                  System.out.println("Salary is = " + net.total);
```

Hierarchical Inheritance

♦ Real time example





Hierarchical - Example

```
Class A {
         int data = 10:
Class B extends A {
Class \subset extends \land {
Class D extends A {
         public static void main(String[] args) {
                  Bb = new B();
                  Cc = new C();
                  Dd = new D();
                  System.out.println("Salary is = " + b.data);
                  System.out.println("Salary is = " + c.data);
                  System.out.println("Salary is = " + d.data);
```

Multilevel Inheritance

- Why multiple inheritance is not supported in java?
 - ♦ Consider a scenario where A, B and C are three classes.
 - ♦ The C class inherits A and B classes.
 - If A and B classes have same method and you call it from child class object, there will be ambiguity to call method of A or B class
- ♦ Since compile time errors are better than runtime errors, java renders compile time error if you inherit 2 classes
- So whether you have same method or different, there will be compile time error now.



Multilevel Inheritance - example

```
♦ Let us look at the following Example:
   class A {
        void display() {
              System.out.println("Hello (A)");
   class B {
        void display() {
              System.out.println("Welcome (B)");
   class C extends A, B {
         public static void main(String[] args) {
              Cc = new C();
              c.display();
```



Hybrid Inheritance

- Any combination of previous three inheritance (single, hierarchical and multi level) is called as hybrid inheritance
- In simple terms, one can say that Hybrid inheritance is a combination of Single and Multiple inheritance
- A hybrid inheritance can be achieved in the java using interfaces



Inheritance – Pros and Cons

♦ Advantages:

- ♦ Application development time is less.
- ♦ Application take less memory.
- ♦ Application execution time is less.
- ♦ Application performance is enhance (improved).
- Redundancy (repetition) of the code is reduced or so that we get consistence results and less storage cost

♦ Disadvantages:

- Base class and child classes are tightly coupled. Hence If you change the code of parent class, it will affect all the child classes
- In class hierarchy many data members remain unused and the memory allocated to them is not utilized. Hence affect performance of your program if you have not implemented inheritance correctly



22/08/19 OCP 14/019 In Sperify

Exercise - 6

- Apply the concepts of Inheritance for
 - ♦ Developing a mobile payment system
 - Design class diagrams and relationship among the objects
 - ♦ Simplify the tasks in mobile payments
 - Define access control mechanism that provides security features for such an application



22/08/19

Assignments / Penalties



- Every Student is expected to complete the assignments and strictly follow a fair Academic Code of Conduct to avoid severe penalties
- ♦ Penalties would be heavy for those who involve in:
 - Copy and Pasting the code
 - Plagiarism (copied from your neighbor or friend in this case, both will get "0" marks for that specific take home assignments)
 - ♦ If the candidate is unable to explain his own solution, it would be considered as a "copied case" !!
 - Any other unfair means of completing the assignments





Assistance

- ♦ You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- You may leave me an email any time (email is the best way to reach me faster)





Thanks ...

