

While folks are joining

Get you laptops ready and login to your **replit** accounts.

We will be coding away in the session!



Crio Sprint : JAVA-2

Session 4



Today's Session Agenda

- Inheritance
- Types of Inheritance
- Final Methods
- Final Class
- Keywords covered
 - **final**
 - **extends**
 - **protected**
 - **super**



Scenario 1

- How many of you regularly shop on Amazon, Flipkart or any other e-commerce website?
- Do you use Credit Card for payment on these sites?
- Ever seen an offer like this?



Limited period offer

OnePlus 9^{5G}
Upgrade to Hasselblad Camera

~~₹49,999~~ **₹45,999[#]**

Additional Exchange bonus of up to ₹7,000
[#]incl. ₹4,000 off with Coupons

HDFC BANK | **₹3,000 OFF*** with
HDFC Bank Credit Cards & EMI

T&C apply

alexia built-in



Scenario 1

- Why do you use Credit Card for?
- What are the basic features provided by any Credit Card?
 - Online/Offline Transactions
 - Emergency Loan
 - Avail Discount Benefits
 - Earn Reward Points
 - ...
- Who can provide you with a credit card?



Scenario 1

- How are these credit cards different from the basic credit card?
- Let's say, for payment with these cards on their respective websites:
 - 5% of the total amount is provided as cashback in Amazon Pay Wallet.
 - 4% is provided as cashback as Flipkart Super Coins.
- Are the above features present in every basic credit card?



Why Inheritance?

- All credit cards have a **common set of features**.
- Some credit cards need to support **specific features**.
- Do we really need to implement all features from scratch for every new credit card type?
 - No, we can **avoid duplicate implementation of these features** across credit cards.
 - Example: SnapDeal can partner with one of the banks quickly for credit card.
 - Embraces **Reusability**
- Reduce development cost and time.
- How do we achieve this in software?
 - We can use **classes and inheritance**.



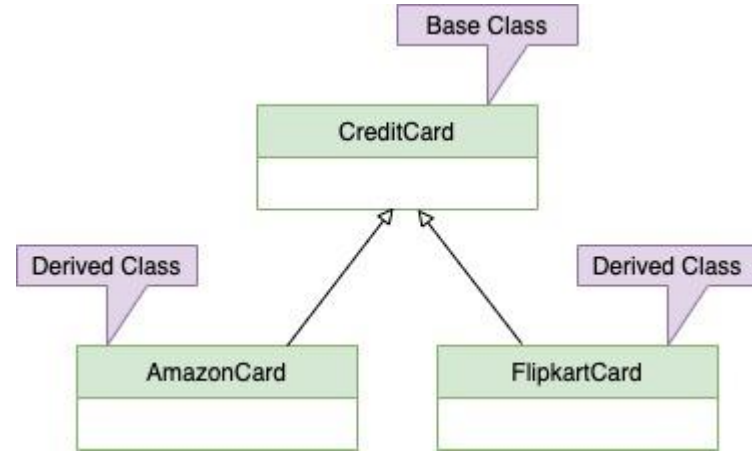
Can you think of other such inheritance scenarios?

- Amazon account and Prime Account
- Feature phone and a Smartphone
- Same Car Model, base variants and higher variants
- YouTube and YouTube Premium Account
- Common Bank functionality and Specific Bank Account functionality
- Crio's BDT/FDT program and Fellowship Program



What is inheritance?

- **Classes can inherit commonly used state and behavior from other classes.**
- CreditCard class has variables & methods which is common to both FlipkartCard and AmazonCard.
- This class will **lend its members** to both of them.
- Here, CreditCard class is a **Superclass**
 - Also called as *Parent class* or *Base class*
- Both FlipkartCard and AmazonCard are **inheriting members** from the Base Class.
- Both these classes are **Subclasses**.
 - Also called as *Child class* and *Derived class*



Inheritance in Java

```
public class CreditCard{  
    private double creditLimit;  
    private Date expiryDate;  
    private String owner;  
    private int rewardPoints;  
  
    public void pay();  
    public void generateBill();  
    public void redeemRewardPoints();  
}
```

Base Class

```
public class FlipkartCard extends CreditCard{  
    public int redeemPointsToSuperCoins();  
}
```

Sub Class

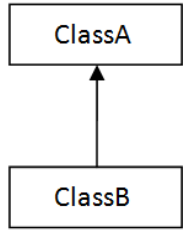
Inherit
properties of a
class

```
public class AmazonCard extends CreditCard {  
    public int  
    redeemPointsToAmazonPayBalance();  
}
```

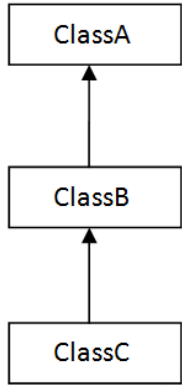
Sub Class



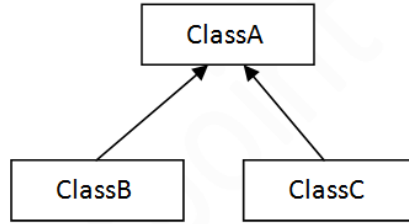
Types of Inheritance



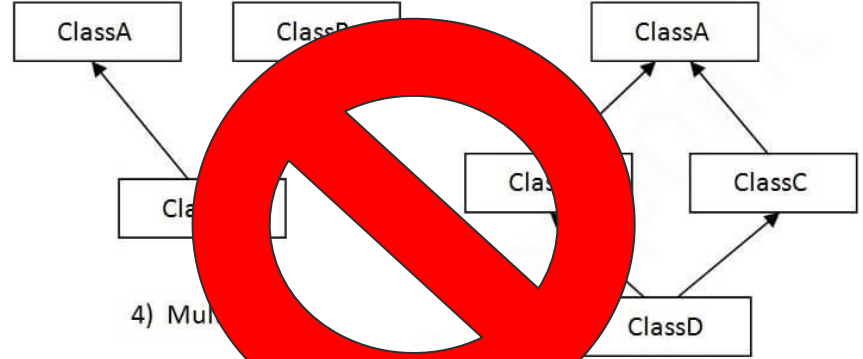
1) Single



2) Multilevel



3) Hierarchical



4) Multiple

5) Hybrid

Try out these inheritance code pieces on repl.it

- [Single Inheritance](#)
- [Multilevel Inheritance](#)
- [Hierarchical Inheritance](#)

In Java, these 2 are not supported through classes (But there is another way to achieve this - interfaces, which will get to in the next session).





- Why is multiple inheritance not supported in Java with classes?
 - Let's consider this scenario.
 - A, B, and C are three classes. The C class inherits A and B classes.
 - If A and B classes have a method with the same name and this method is invoked from C, which inherited method should be called? Method from A or Method from B?
 - Java renders compile-time error if you inherit 2 classes.

```
class A{  
    void msg(){System.out.println("Hello");}  
}  
  
class B{  
    void msg(){System.out.println("Welcome");}  
}  
  
class C extends A,B{  
  
    public static void main(String args[]){  
        C obj=new C();  
        obj.msg(); //Now which msg() method  
        would be invoked?  
    }  
}
```



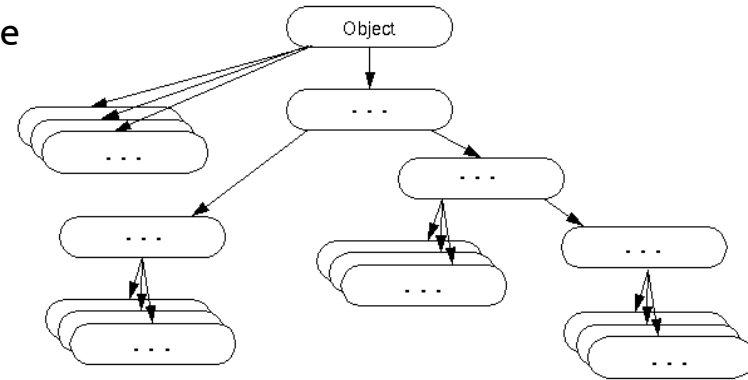


- Can constructors be inherited in Java?
 - A constructor cannot be inherited, as the subclasses always have a different name.



Godfather class in Java

- Do you know which is the superclass for all classes in Java?
- Why?
- The **Object** Class in Java
- Root of inheritance hierarchy.
- Its methods available to all Java classes.
- Present in java.lang package.
- Every class in Java is directly or indirectly derived from the Object class.



protected keyword

- An access modifier that grants access of a class members to
 - Classes belonging to the same package as the given class

```
package p1;  
public class Person {  
    protected String name;  
}
```

```
package p1;  
public class Employer {  
    void hireEmployee() {  
        Person p = new Person();  
        p.name = "Nam"; // access protected variable directly  
    }  
}
```

- Subclass of the given class

```
package p2;  
import p1.Person;  
class Employee extends Person {  
    void doStuff() {  
        name = "Bob";  
    }  
}
```

```
package p2;  
import p1.Person;  
class AnotherEmployer {  
    void hire() {  
        Person p = new Person();  
        // compile error, cannot access protected variable  
        // from different package  
        p.name = "Nam";  
    }  
}
```



How would you do this in Java?

- A unique mobile (the iPhone Yoda model) has been introduced!
 - Apple wants this to be a one time model that no one replicates.
 - How would they do that?
- There is a counter method which increments count, every time a new instance of the class is created
 - The developer doesn't want anyone who uses this counter to modify this logic.
 - How would they do that?
- Answer - The **final** keyword



Activity 1 - Find a Bug

```
public class Calculator{  
  
    public int add(int a, int b){  
        return a + b;  
    }  
    public int subtract(int a, int b){  
        return a - b;  
    }  
    public int multiply(int a, int b){  
        return a * b;  
    }  
    public int divide(int a, int b){  
        return a / b;  
    }  
}
```

```
public class ScientificCalculator  
extends Calculator{  
  
    @Override  
    public int add(int a, int b){  
        return a - b;  
    }  
    @Override  
    public int subtract(int a, int b){  
        return a + b;  
    }  
    public int square(int a){  
        return a * a;  
    }  
    public double divide(int a){  
        return Math.sqrt(a);  
    }  
    // several other methods  
}
```

- What's the issue with ScientificCalculator Class?
- How can we stop overriding the methods?
 - Mark the methods as **final**.
- A method marked as final **cannot be overridden**.



Curious Cats



- Can we inherit a String Class? Why or Why not?
- It is an immutable class.
 - How can we make a class immutable?
 - Mark the class as final.

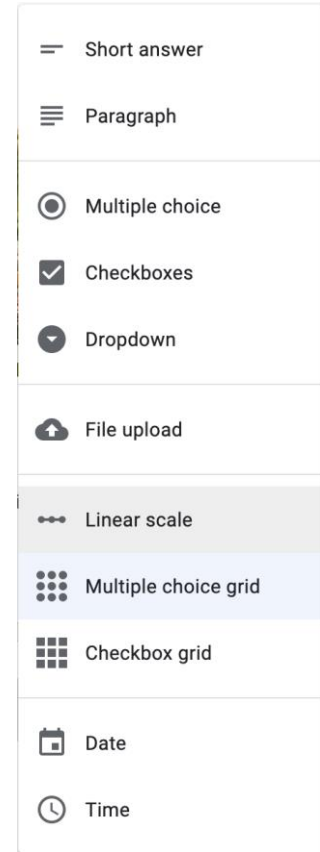
```
final class String{}
```

- What are the other benefits of final classes?
 - Prevent further inheritance
 - It cannot be extended



Activity 2 - Simple Quiz

- Google form is a popular tool which allows you to create surveys, quiz, and much more.
- It provides different ways to gather information.
- In this activity, we will be implementing a quiz application with these kind of questions:
 - Short Answer Questions
 - Multiple Choice Questions
- [Simple Quiz](#)



A screenshot of the Google Forms question type selection menu. The menu is a vertical list of options, each with an icon and a label. The options are: 'Short answer' (horizontal lines icon), 'Paragraph' (three horizontal lines icon), 'Multiple choice' (radio button icon), 'Checkboxes' (checkmark icon), 'Dropdown' (downward arrow icon), 'File upload' (cloud with up arrow icon), 'Linear scale' (horizontal line with dots icon), 'Multiple choice grid' (3x3 grid of dots icon, highlighted with a blue background), 'Checkbox grid' (3x3 grid of squares icon), 'Date' (calendar icon), and 'Time' (clock icon).

- Short answer
- Paragraph
- Multiple choice
- Checkboxes
- Dropdown
- File upload
- Linear scale
- Multiple choice grid
- Checkbox grid
- Date
- Time



Activity 2.1 - Short Answer Support

Short Answer Question

- What are the fields you can identify from this image?
 - question
 - answer
- What are the behaviours you think would be required for the fields?
 - Setters and getters
 - Check correct answer
 - Display the question
- Go to Activity 2.1 and implement the above defined requirements.
 - [Activity 2.1 Instructions](#)

What is your Email ID? *

Short answer text



Activity 2.2 - Multiple Choice Question Support

Multiple Choice Question

- How does a Multiple Choice Question differ from Short Answer?
 - It store choices in addition to the question
- What are the extra fields you think might be required?
 - List of choices
- What are the extra behaviours you think would be required for the fields?
 - Add choices in the list
 - Display the MCQ question (Override)
- Is it possible to inherit the remaining fields/behaviour from Short Answer?
- Go to Activity 2.2 and implement the above defined requirements.
 - [Activity 2.2 Instructions](#)

Your first question? *

- ☐ Option 1
- ☐ Correct answer
- ☐ Option 3
- ☐ Option 4



Simple Quiz - New things we used

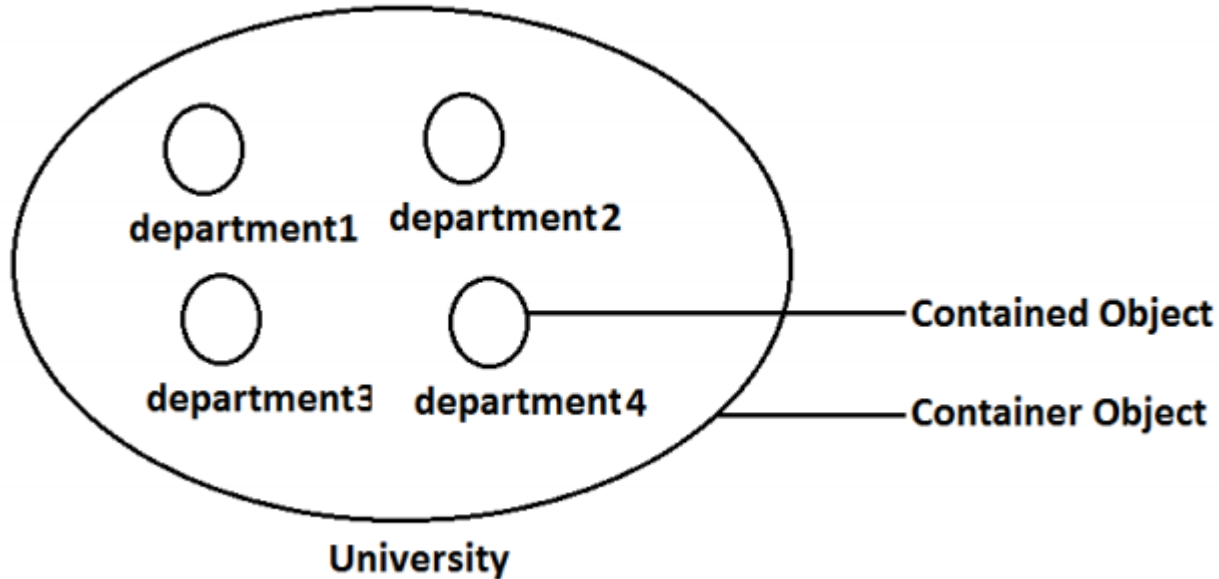
- **super keyword**
 - refers to superclass (parent) objects.
- **protected access modifier**
 - An access modifier used for attributes, methods and constructors, making them accessible in the same package and subclasses.
- **Method overriding**
 - A child class can give its own implementation to a method which is already provided by the parent class.



Curious Cats



- How is composition different than inheritance?



- [Inheritance vs. Composition. And which should you use?](#)
| by Brian | Better Programming



Take home exercises for the session

- You will explore **Inheritance** with this real world scenario in the following Byte:
 - [Inheritance Byte - Crio.do](#)
- Complete the Quiz Activity (details on the google site) for Session 4.
 - [Java 2 Session 4 Quiz](#)
- Try to finish this before the next session on Thursday, 7:30 pm.

These details are also available on the site.



Feedback

Thank you for joining in today.

We'd love to hear your thoughts and feedback - [Feedback for JAVA-2 Session](#)



Further Reading

- [Java Protected Keyword - Javatpoint](#)



References

- [Oracle Docs - Access Control](#)



Thank you

