Object Oriented Development with Java

(CT038-3-2 and Version VC1)

Inheritance

Topic & Structure of the lesson

- Introduction
- Single inheritance
- Multiple inheritance
- Multi-level inheritance
- Inheriting Methods

Learning outcomes

- At the end of this lecture you should be able to:
- Implement inheritance
- Understand how to inherit and override superclass methods

Key terms you must be able to use

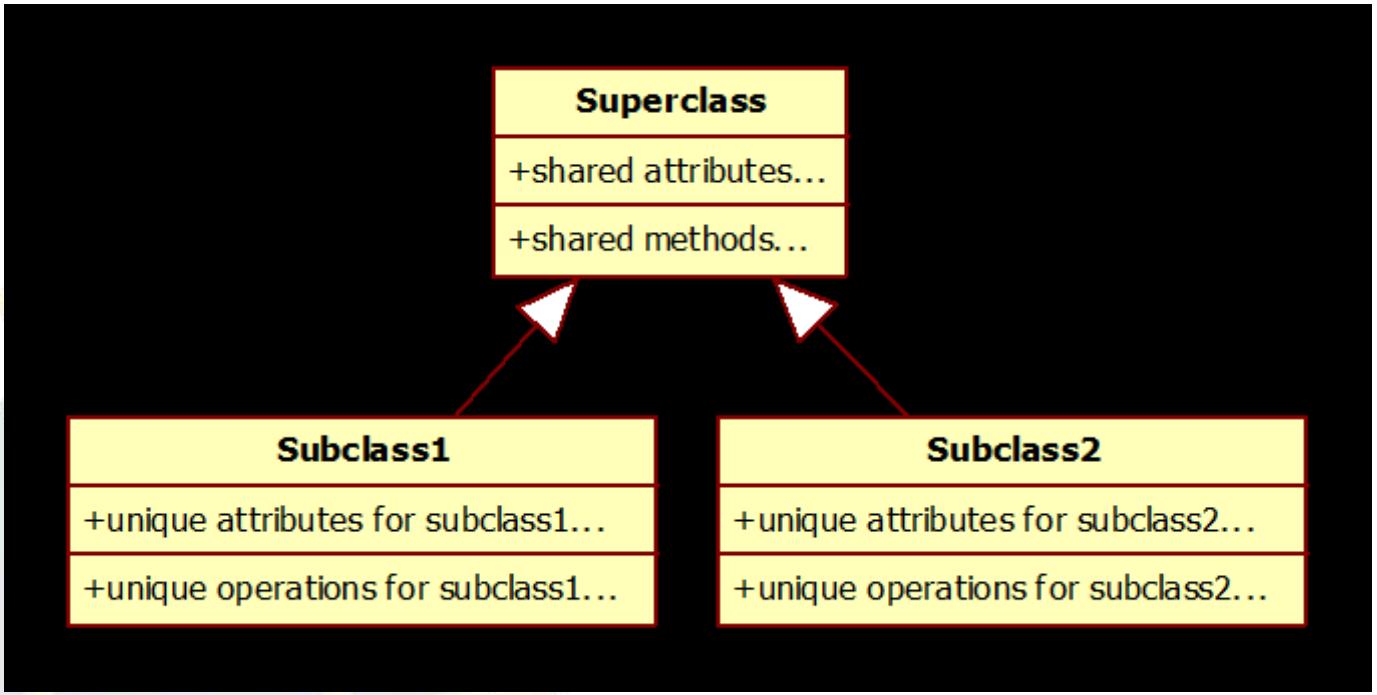
If you have mastered this topic, you should be able to use the following terms correctly in your assessments:

- Inheritance
- Single inheritance
- Multilevel Inheritance
- Overriding

Introduction

- Allows methods and fields to be shared amongst different objects
- Eg: Vehicles include bicycles, skateboards, cars, jets (etc.) share some common features and some that are unique to each particular type
- Shared features are collected in a single class known as the parent or *superclass*
- Unique features are separated into the child or *subclasses*

Introduction



- subclasses inherit all the variables and methods of their parent classes

Introduction

- Inheritance: extend existing classes by adding methods and fields
- mechanism of deriving a new class from an old one
  - old class = base class / super class / parent class
  - new class = sub class / derived class / child class

Syntax : Inheritance

```
class SubclassName extends SuperclassName
  {
     methods
     instance fields
  }
```

Example

```
class SavingsAccount extends BankAccount
{
  new methods
  new instance fields
}
```

Example

```
public class BankAccount {
   private double balance;

public BankAccount(){
   balance = 0; }
public BankAccount (double initialBalance) {
  balance = initialBalance; }
 public void deposit (double amount) {
  balance = balance + amount; }
public void withdraw (double amount) {
  balance = balance - amount;}

//more methods here ie. getBalance() and transfer()
}
```

Example

```
public class SavingsAccount extends BankAccount
{
   public SavingsAccount(double rate)
   {
       interestRate = rate;
   }
   public void addInterest()
   {   double interest = getBalance() * interestRate / 100;
       deposit(interest);
   }
   private double interestRate;
}
```

Example

```
SavingsAccount collegeFund = new SavingsAccount(10);
//Savings account with 10% interest
collegeFund.deposit(500);
// OK to use BankAccount method with SavingsAccount
object
```

- SavingsAccount automatically inherits all methods and instance fields of BankAccount
- A superclass has state and behavior, and the subclass inherits them.
- One advantage of inheritance is code reuse

Forms of Inheritance

- Inheritance may take the following forms:
  - single inheritance
  - multiple inheritance
  - multi-level inheritance

Single Inheritance

- Only one superclass

```
class Employee //superclass
{

}

class Manager extends Employee //subclass
{

}
```

Multiple Inheritance

- Several superclasses
- Not explicitly supported by Java
- Cannot use keyword extends
- Need to use Interface – keyword *implements*
  - will be discussed in another lecture

Multiple Inheritance

- Format

```
class Abc extends One implements Two
{
  ....
}
class One
{
  ...
}
class Two
{
```

Example: Hierarchy of Bank Accounts

- Consider a bank that offers its customers the following account types:
- Checking account:
  - no interest
  - small number of free transactions per month
  - additional transactions are charged a small fee
- Savings account:
  - earns interest that compounds monthly

Example: Hierarchy of Bank Accounts

- Inheritance hierarchy:
  - All bank accounts support the getBalance method
  - All bank accounts support the deposit and withdraw methods, but the implementations differ
  - Checking account needs a method deductFees
  - savings account needs a method addInterest

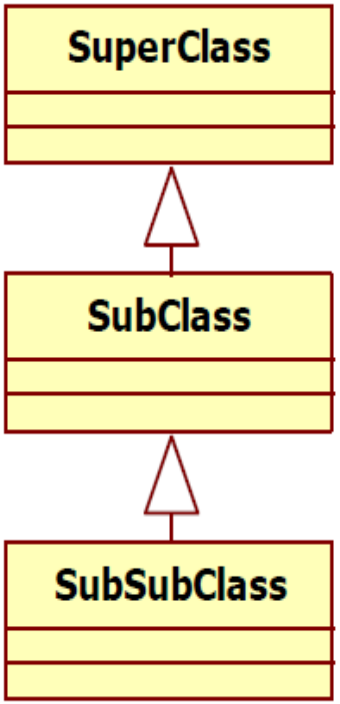
Multi-level Inheritance

- Multiple levels of sharing
- Supervisor inherits from both Manager and Employee

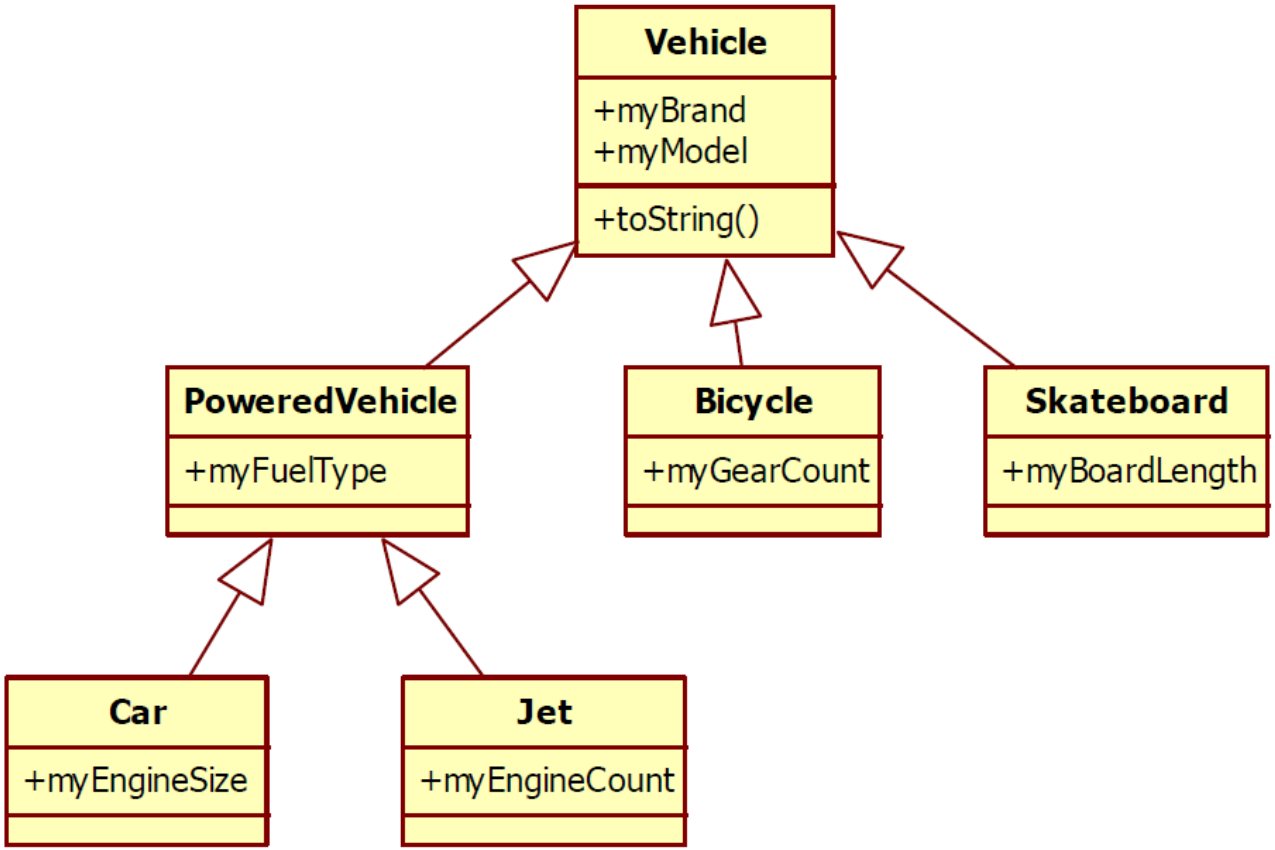
```
class Employee //superclass
{
}

class Manager extends Employee //subclass
{
}

class Supervisor extends Manager //subsubclass
{
}
```



Multi-level Inheritance Class Diagram



Override Method

- Supply a different implementation of a method that exists in the superclass
- Must have same name and same parameter types
- If method is applied to an object of the subclass type, the overriding method is executed

Override Method

```
class Superclass {

      // other features of the superclass...

      public int superClassMethod() {
            // do something...
            return 1;
      }
}

class Subclass extends Superclass {

      // other features of the subclass...

      public int superClassMethod() {
            // do something different...
            return 2;
      }
}
```

Example: Implementing the CheckingAccount Class

```
public class CheckingAccount extends BankAccount
{
   public void deposit(double amount){ . . . }
   public void withdraw(double amount) { . . . } public
   void deductFees() { . . . } // new method
   private int transactionCount; // new instance field
}
```

Example: Implementing the CheckingAccount Class

- Each CheckingAccount object has two instance fields:

```
balance (inherited from BankAccount)
transactionCount (new to CheckingAccount)
```

Example: Implementing the CheckingAccount Class

- Can apply four methods to CheckingAccount objects:

```
getBalance() (inherited from BankAccount)
deposit(double amount) (overrides BankAccount method)
withdraw(double amount) (overrides BankAccount method)
deductFees() (new to CheckingAccount)
```

Quick Review Questions

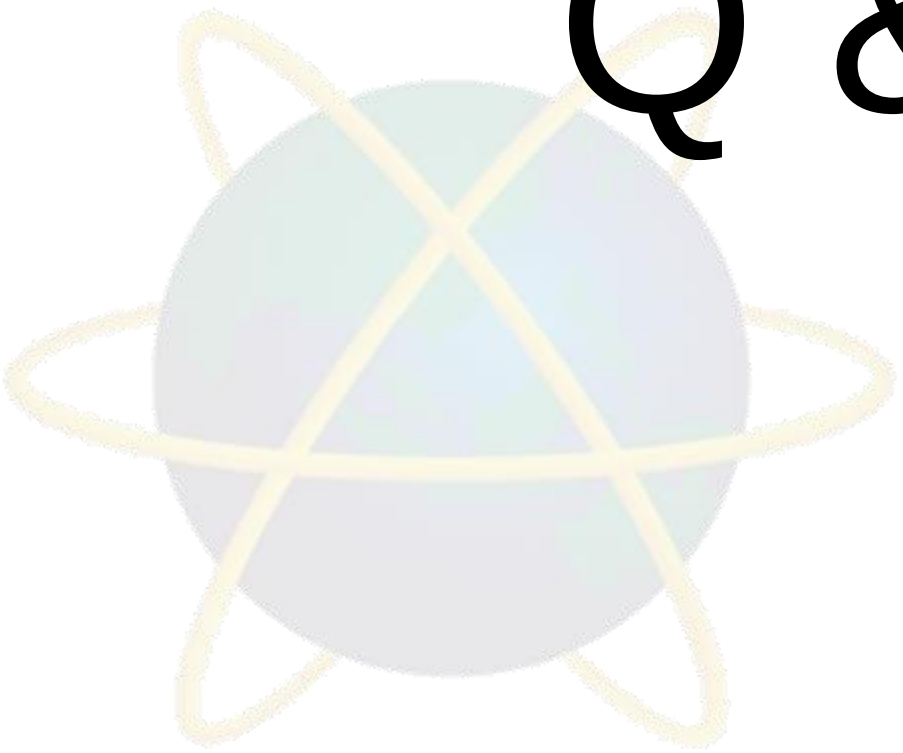
- What is inheritance?
- What are the three forms of inheritance?
- Show an example of overriding a superclass

Summary of Main Teaching Points

A · P · U
ASIA PACIFIC UNIVERSITY
OF TECHNOLOGY & INNOVATION

- Inheritance
- Single Inheritance
- Multilevel Inheritance
- Override

Question and Answer Session



Q & A

Next Session

- Polymorphism
- Abstract Classes & Methods
  Example
- Interfaces
  Example
- Interfaces vs. Abstract Classes