GUIDED PRACTICE

Class: *CS 2012 - Introduction to Programming II*
Date assigned: TBA
Date due: TBA
Time estimate to complete this assignment: *80 minutes*

Overview/Introduction

Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class. Inheritance supports the concept of "reusability", i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

Java supports class reuse through inheritance and composition. In this two-part miniseries we'll focus on inheritance, one of the fundamental concepts of object-oriented programming. Students will learn how to use the "extends" and "implements" keyword to derive a child class from a parent class, invoke parent class constructors and methods, and override methods. Students also will tour "java.lang.Object" and its methods. "Object" is Java's ultimate superclass, from which every other class inherits.

Learning Objectives

Basic objectives

1. Define what inheritance is.
2. Describe how inheritances are implemented in Java:
    a. subclass
    b. superclass
    c. interface
3. Define the keywords:
    a. extends
    b. implements
4. Define the keywords:
    a. public
    b. private
    c. protected

Advanced objectives

1. Be able to identify subclasses and superclasses in the real-world.
2. Be able to describe the relationships of among the subclasses and superclasses.
3. Be able to write a program based on Java inheritance, such as:

a. base class: person
b. subclass: student
c. subclass: faculty

Preparatory Activities and Resources:

Step 1: Watch the video “Introducing-inheritance”
“https://www.lynda.com/Java-tutorials/Introducing-inheritance/375490/415281-4.html”
15 minutes

Step 2: Watch the video “Super and Sub Classes (Java)”,
“https://www.youtube.com/watch?v=0PPKccntohM”
15 minutes

Step 3: Watch the video “Extending Classes (Java)”,
“https://www.youtube.com/watch?v=GDG-wzEZW8E”
15 minutes

Step 4: Watch the video “Java extends vs implements”
“https://www.youtube.com/watch?v=BtCVX79KGPw”
20 minutes

Step 5: Watch the video ”Protected Access (J)”
“https://www.youtube.com/watch?v=1MM1F9xK44M”
15 minutes

Exercises: Please complete by __TBA___.

- The exercises for this lesson are found on the Google Form at: Java Inheritance. Work out these exercises in your own notes so you’ll have a record of your work and then take the quiz. Remember your work is graded Pass/Fail on the basis of completeness, effort, and timeliness only.

Questions?

- Students can contact faculty at Moodle message center.

Lesson Plan for Inheritance in Java

CS 2012 - Introduction to Programming II

Jiang Guo

Department of Computer Science

California State University Los Angeles

Lesson: Inheritance in Java

Timeframe: Approximately 50 minutes

Materials needed:

- **hardware –** Windows or Linux or Mac Computers
- **software –** Java, Eclipse or any editor

Objectives:

- **Basic:**
  - Define what inheritance is.
  - Describe how inheritances are implemented in Java:
    a. subclass
    b. superclass
    c. interface
  - Define the keywords:
    a. extends
    b. implements
  - Define the keywords:
    a. public
    b. private
    c. protected

- **Advanced:**
  - Be able to identify subclasses and superclasses in the real-world.
  - Be able to describe the relationships of the subclasses and superclasses.
  - Be able to write a program based on Java inheritance, such as:
    a. base class: person
    b. subclass: student
    c. subclass: faculty

Background:

Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class. Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

Introduction to the Lesson:

Java supports class reuse through inheritance and composition. In this two-part miniseries we'll focus on inheritance, one of the fundamental concepts of object-oriented programming. Students will learn how to use the “extends” and “implements” keyword to derive a child class from a parent class, invoke parent class constructors and methods, and override methods. Students also will tour ”java.lang.Object” and its methods. ”Object” is Java's ultimate superclass, from which every other class inherits.

Procedure:

Pre-Class Individual Space Activities and Resources [80 minutes]:

Steps	Purpose	Estimated Time	Learning Objective
Step 1: Watch the video “Introducing-inheritance” “https://www.lynda.com/Java-tutorials/Introducing-inheritance/375490/415281-4.html”	Introduce students to concept of inheritance.	15 min.	#1, (Basic)
Step 2: Watch the video “Super and Sub Classes (Java)”, “https://www.youtube.com/watch?v=0PPKccntohM”	Introduce students to concepts of subclass and superclass.	15 min.	#2, (Basic)

Step 3: Watch the video “Extending Classes (Java)”, “https://www.youtube.com/watch?v=GDG-wzEZW8E”	Introduce students to concepts of extends and implements.	15 min	#3, (Basic)
Step 4: Watch the video “Java extends vs implements” “https://www.youtube.com/watch?v=BtCVX79KGPw”	Help students to distinguish concepts of extends and implements.	20 min	#3, (Basic)
Step 5: Watch the video ”Protected Access (J)” “https://www.youtube.com/watch?v=1MM1F9xK44M”	Introduce students to concepts of public, private, and protected.	15 min	#4, (Basic)

In-Class Group Space Activities and Resources [50 minutes]:

Steps	Purpose	Estimated Time	Learning Objective
• Step 1: Open Minutes: ✓ Students in groups draw a picture to show the definition of the inheritance and give three examples of the inheritance in different areas. (5 minutes) (Computer) ✓ Students in groups describe the difference of: extends and implements. (5 minutes) (Computer) ✓ Students in groups describe the difference among: public, private, and protected. (5 minutes) (Computer)	• Clear up any general confusion or misconceptions. • Review and reinforce the material introduced in the individual space.	15 min.	All Basic LOs

• Step 2: Main Activities: ✓ Draw the diagram to describe the relationships among: person, student, faculty. (5 minutes) (Computer) ✓ Based on the diagram, write a simple framework of the program of: person, student, faculty by using: a. extends or/and b. implements keywords. (10 minutes) (Computer) ✓ Extend the framework into a program of: person, student, faculty. Add data attributes: a. age, b. id, c. cin, d. salary, e. phone number, (10 minutes) (Computer)	• Have students apply the concepts they learned in the individual space to a practical case study. • Improve student critical thinking ability • Provide students deep learning opportunities.	25 min.	All Advanced LOs
• Step 3: Main Activities: ✓ Review the program and identify the improvement space (better structure, balanced design) of the program (10 minutes) (Computer)	Provide students opportunities to applying concepts to more difficult case studies.	10 min.	All Advanced LOs

Closure/Evaluation [5 minutes]:

Describe the upcoming homework assignment and discuss how it relates to the group space activities that were completed in class.

Analysis:

Inheritance is a programming construct that software developers use to establish is-a relationships between categories. Inheritance enables us to derive more-specific categories from more-generic ones. The more-specific category is a kind of the more-generic category. For

example, a checking account is a kind of account in which you can make deposits and withdrawals. Similarly, a truck is a kind of vehicle used for hauling large items.

Students in this class are primarily majoring in the computer science and they are trained to write simple programs. The in-class group space activity is designed to allow students to apply the general concepts they learn in the pre-class individual space activities in design and develop real Java programs. Eclipse software is extremely powerful software development environment. Students in the class will focus on applying the concepts to write Java programs and they will reinforce their skills learned in classroom by doing the post-class individual space activity.

Post-Class Individual Space Activities:

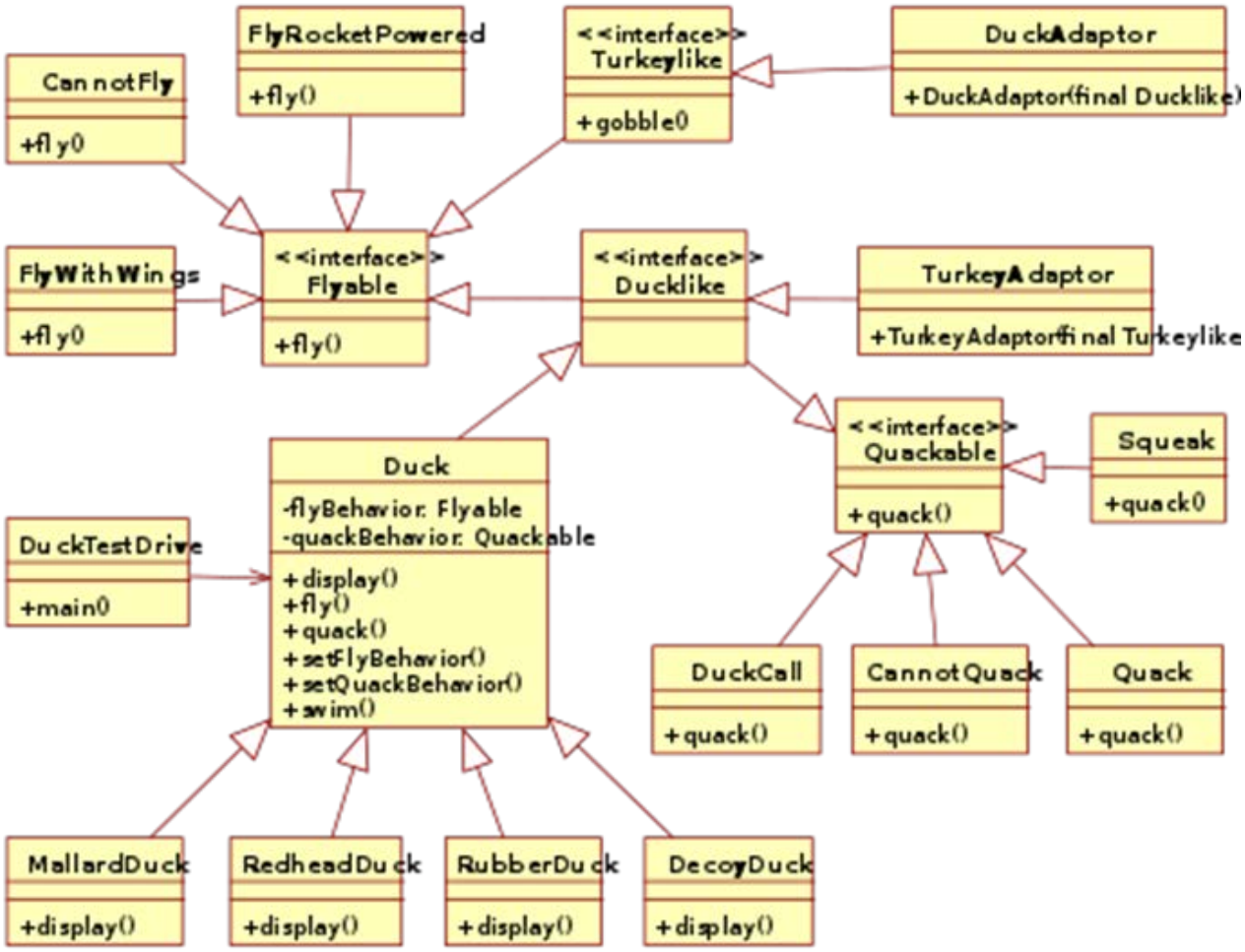
Students will apply their knowledge of Java inheritance in a homework assignment. Specifically, they will design a program to implement the Java inheritance. Their assignment will be to use an Eclipse to write a complex program based on UML diagram and create all the relationships between the classes.

Connections to Future Lesson Plan(s):

The topic that comes next is polymorphism means to process objects differently based on their data type. Polymorphism is based on Inheritance. Polymorphism means: one method with multiple implementations, for a certain class of action can be decided at runtime depending upon the situation to use the correct implementation. This can be implemented by designing a generic interface, which provides generic methods for a certain class of action and there can be multiple class inheritance, which provides the implementation of these generic methods.

Appendix A: Homework Assignment

Design a program to implement inheritance as describe in below diagram.



Appendix B: Program Example

Relating classes through inheritance

The `extends` keyword specifies a parent-child relationship

```
class Vehicle
{
   // member declarations
}
class Car extends Vehicle
{
   // inherit accessible members from Vehicle
   // provide own member declarations
}
class Account
{
   // member declarations
}
class SavingsAccount extends Account
{
   // inherit accessible members from Account
   // provide own member declarations
}
```

An `Account` parent class

```
class Account
{
   private String name;

   private long amount;
```

```
   Account(String name, long amount)
   {
      this.name = name;
      setAmount(amount);
   }

   void deposit(long amount)
   {
      this.amount += amount;
   }

   String getName()
   {
      return name;
   }

   long getAmount()
   {
      return amount;
   }

   void setAmount(long amount)
   {
      this.amount = amount;
   }
}
```

A `SavingsAccount` child class extends its `Account` parent class

```
class SavingsAccount extends Account
{
   SavingsAccount(long amount)
   {
      super("savings", amount);
```

```
    }
}
```

A `CheckingAccount` child class extends its `Account` parent class

```
class CheckingAccount extends Account
{
   CheckingAccount(long amount)
   {
      super("checking", amount);
   }

   void withdraw(long amount)
   {
      setAmount(getAmount() - amount);
   }
}
```

`AccountDemo` demonstrates the account class hierarchy

```
class AccountDemo
{
   public static void main(String[] args)
   {
      SavingsAccount sa = new SavingsAccount(10000);
      System.out.println("account name: " + sa.getName());
      System.out.println("initial amount: " + sa.getAmount());
      sa.deposit(5000);
      System.out.println("new amount after deposit: " + sa.getAmount());

      CheckingAccount ca = new CheckingAccount(20000);
      System.out.println("account name: " + ca.getName());
      System.out.println("initial amount: " + ca.getAmount());
      ca.deposit(6000);
```

```
      System.out.println("new amount after deposit: " + ca.getAmount());
      ca.withdraw(3000);
      System.out.println("new amount after withdrawal: " + ca.getAmount());
   }
}
```

Declaring a `print()` method to be overridden

```
class Vehicle
{
   private String make;
   private String model;
   private int year;

   Vehicle(String make, String model, int year)
   {
      this.make = make;
      this.model = model;
      this.year = year;
   }

   String getMake()
   {
      return make;
   }

   String getModel()
   {
      return model;
   }

   int getYear()
   {
      return year;
```

```
    }

    void print()
    {
        System.out.println("Make: " + make + ", Model: " + model + ", Year: " +
                           year);
    }
}
```

Overriding `print()` in a `Truck` subclass

```
class Truck extends Vehicle
{
    private double tonnage;

    Truck(String make, String model, int year, double tonnage)
    {
        super(make, model, year);
        this.tonnage = tonnage;
    }

    double getTonnage()
    {
        return tonnage;
    }

    void print()
    {
        super.print();
        System.out.println("Tonnage: " + tonnage);
    }
}
```

Appendix C: Program Example

In below example of inheritance, class Bicycle is a base class, class MountainBike is a derived class which extends Bicycle class and class Test is a driver class to run program.

```
//Java program to illustrate the
// concept of inheritance

// base class
class Bicycle
{
    // the Bicycle class has two fields
    public int gear;
    public int speed;

    // the Bicycle class has one constructor
    public Bicycle(int gear, int speed)
    {
        this.gear = gear;
        this.speed = speed;
    }

    // the Bicycle class has three methods
    public void applyBrake(int decrement)
    {
        speed -= decrement;
    }

    public void speedUp(int increment)
    {
        speed += increment;
    }

    // toString() method to print info of Bicycle
    public String toString()
    {
        return("No of gears are "+gear
                +"\n"
                + "speed of bicycle is "+speed);
    }
}

// derived class
class MountainBike extends Bicycle
{

    // the MountainBike subclass adds one more field
    public int seatHeight;

    // the MountainBike subclass has one constructor
    public MountainBike(int gear,int speed,
                        int startHeight)
```

```
    {
        // invoking base-class(Bicycle) constructor
        super(gear, speed);
        seatHeight = startHeight;
    }

    // the MountainBike subclass adds one more method
    public void setHeight(int newValue)
    {
        seatHeight = newValue;
    }

    // overriding toString() method
    // of Bicycle to print more info
    @Override
    public String toString()
    {
        return (super.toString()+
                "\nseat height is "+seatHeight);
    }

}

// driver class
public class Test
{
    public static void main(String args[])
    {

        MountainBike mb = new MountainBike(3, 100, 25);
        System.out.println(mb.toString());

    }
}
```