

Topic 7

Inheritance and Introduction to Event-driven Programming

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

ICT167 Principles of
Computer Science

© Published by Murdoch University, Perth, Western Australia, 2020.

Assignment Project Exam Help

This publication is copyright. Except as permitted by the Copyright Act no part of it may in any form or by any electronic, mechanical, photocopying, recording or any other means be reproduced, stored in a retrieval system or be broadcast or transmitted without the prior written permission of the publisher

Objectives

- Define **inheritance** and **polymorphism**
- Be able to give examples of uses of inheritance
[Assignment Project Exam Help](#)
- Be able to use the correct terminology for inheritance (**base** class and **derived** class)
<https://powcoder.com>
- Understand inheritance between Java classes
- Explain the concept of **overriding** of methods
- Distinguish between **overriding** and **overloading**

Objectives

- Explain the use of the term **super** in a constructor
- Understand a **class hierarchy**
- Know that a derived class object can have a super class <https://powcoder.com>
- Be able to define and use **derived classes** in Java
- Understand the concept of multiple inheritance
- Understand Java **interfaces**

Objectives

- Understand the basics of event-driven programming
- Explain the term GUI
- Give a brief description of the Java Swing event-driven programming
- Be able to determine and explain the behaviour of simple Java GUI programs

Reading Savitch: Chapters 8.1, 8.2, 8.3 and Chapter 13 (see textbook website)

Inheritance

- Inheritance enables us to define a new class based on a (general) class that already exists [Assignment](#) [Project](#) [Exam](#) [Help](#)
- The new class ~~[will be similar](https://powcoder.com)~~ to the existing class, it ~~will be able to use all~~ [Add WeChat](#) [powcoder](#) the facilities of the existing class, but will have some new characteristics
- This makes programming easier, because you can build upon your previous work instead of starting out from scratch

Inheritance

- In Java it is easy to code the more specialized class without having to re-write any of the code which it inherits from the more general class
<https://powcoder.com>
Add WeChat powcoder
- Inheritance is a powerful and very useful feature of OOP
 - Graphical user interfaces (GUIs) define each visual component by using inheritance with a "toolkit" of basic components

Inheritance

- For example in the libraries:
 - Buttons inherit from Components
 - Labels inherit from Components
 - FileNotFoundException inherits from Add WeChat powcoder
IOException
 - HttpURLConnection inherits from URLConnection
 - Time inherits from Date
 - Set inherits from Collection

Inheritance

- Example in possible applications,
 - ReferenceBook inherits from LibraryBook
 - UnderGraduateStudent inherits from Student
 - Secretary inherits from Employee
 - CreditCardCustomer inherits from Customer

Terminology

- The class that is used as a basis for defining a new class is called the base class (or super class or parent class)
Add WeChat powcoder
- The new class based on the base class is called a derived class (or subclass or child class)
- We say, the derived class inherits from the base class

Terminology

- In Java, (unlike with humans) child classes inherit characteristics from just one parent
 - This is called single inheritance
 - Some languages allow child classes to inherit from more than one parent
 - This is called multiple inheritance

Terminology

- With multiple inheritance, it is sometimes hard to tell which parent class will contribute what characteristics to the child class
 - Assignment Project Exam Help
<https://powcoder.com>
 - Add WeChat powcoder
- Java avoids these problems by using single inheritance

Example: Base Class

```
// A Base Class: Person.java (from Savitch chapter 8)

public class Person {
    private String name; // instance variable
    public Person() { // constructor
        name = "No name yet.";
    }
    // another constructor
    public Person(String initialValue) {
        name = initialValue;
    }
    public String getName() { // get method
        return name;
    }
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Example: Base Class

```
// set method
public void setName(String newName) {
    name = newName;
}
public void writeOutput(){ // output method
    System.out.println("Name: " + name);
}
// equal method
public boolean sameName(Person otherPerson) {
    return (this.name.equalsIgnoreCase(
        otherPerson.name));
}
} // end class Person
```

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

Example: Derived Class

```
// A Derived Class: Student.java from Savitch chapter 8  
public class Student extends Person {  
    private int studentNumber; // instance variable  
    public Student() // default constructor  
        // call to default constructor of super class Person  
        super();  
    studentNumber = 0; // Indicating no number yet  
}
```

Example: Derived Class

```
// Another constructor
public Student(String initialName,
               int initialStudentNumber) {
    // call to other constructor of super class Person
    super(initialName);
    studentNumber = initialStudentNumber;
}

public void reset(String newName,
                  int newStudentNumber) {
    // call to the super class method
    setName(newName);
    studentNumber = newStudentNumber;
}
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Example: Derived Class

```
public int getStudentNumber() {  
    return studentNumber;  
}  
public void setStudentNumber(int  
                           newStudentNumber) {  
    studentNumber = newStudentNumber;  
}  
public void writeOutput() {  
    System.out.println("Name: " + getName());  
    System.out.println("Student Number : " +  
                      studentNumber);  
}
```

Assignment Project Exam Help

<https://powcoder.com>

newStudentNumber) {

studentNumber = newStudentNumber;

}

public void writeOutput() {

System.out.println("Name: " + getName());

System.out.println("Student Number : " +

studentNumber);

Example: Derived Class

```
public boolean equals(Student otherStudent) {  
    return (this.name == otherStudent.name)  
        && (this.studentNumber ==  
            otherStudent.studentNumber);  
}  
// end class Student
```

Assignment Project Exam Help
Add WeChat powcoder
<https://powcoder.com>

Example: Client Class

```
// InheritanceDemo.java - a client program
public class InheritanceDemo {
    public static void main(String[] args) {
        Student s = new Student();
        s.writeOutput();
        // setName is inherited from the Person class
        s.setName("Jason Bourne");
        s.setStudentNumber(12345678);
        s.writeOutput();
```

Example: Client Class

```
Student s1 = new Student("James Bond", 007);
s1.writeOutput();
if (s.equals(s1))
    System.out.println("Same");
else
    System.out.println("Not Same");
}

} // end class InheritanceDemo
```

Example: Output

```
/* OUTPUT  
Name: No name yet.  
Student Number : 0  
Name: Jason Bourne  
Student Number: 12345678  
Name: James Bond  
Student Number: 7  
Not Same  
*/
```

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

Overriding Methods and super Constructors

- In the above example (consisting of two useful classes and a client) notice the following:
 - Person is the base class (or super class)
 - Student is derived from the base class
 - We also say, Student **inherits** from Person, or, Student **extends** Person
 - The most important thing to note is that each Student object (like any other Person object) has a name and has methods such as setName (...) and getName () available to it
 - This is inheritance

Overriding Methods and super Constructors

- Each Student object also has an extra instance variable `studentNumber`
[Assignment](#) [Project](#) [Exam](#) [Help](#)
- An object of type `Student` has the following members in it:
<https://powcoder.com>
[Add](#) [WeChat](#) [powcoder](#)

Overriding Methods and super Constructors

Member	Explanation
name	inherited from Person
studentNumber	defined in Student
setName(...)	inherited from Person
getName()	inherited from Person
sameName(...)	inherited from Person
reset(...)	defined in Student
getStudentNumber()	defined in Student
setStudentNumber (...)	defined in Student
writeOutput()	redefined in Student
equals(...)	defined in Student

Overriding Methods and super Constructors

- Also note that where the derived class has a method with exactly the same name and the same number, order and types of parameters as a method in the base class then the derived class method will be used for a derived class object
 - Eg: the method `writeOutput()` in Student class above
 - This is called **overriding**

Overriding Methods and super Constructors

- Also note that since a Student is a type of Person then a new Student object must be set up properly as a Person first
- In general, a constructor for the super class must be called as part of the activity of a constructor for the derived class. If you do not specify which super constructor to call by writing `super(arg1, ..., argn)` in the derived class constructor then the default constructor is called automatically

Overriding Methods and super Constructors

- The class definition for Person has two constructors, one of which will initialise the member data of Person objects
<https://powcoder.com>
- The class Student also has two constructors that initialise the data of Student objects
- The second constructor for class Student looks like following:

Overriding Methods and super Constructors

```
// another constructor
public Student(String initialName,
                int initialStudentNumber) {
    // call to the constructor of superclass
    super(initialName);
    // initialise the member that only Student has
    studentNumber = initialStudentNumber;
}
```

Assignment Project Exam Help

<https://powcoder.com>
Add WeChat powcoder

Overriding Methods and super Constructors

- The statement `super(initialName)` invokes the ~~Assignment Project Exam's Help~~ super class's constructor to initialize some of the data
<https://powcoder.com>
- The next statement initializes the member ~~Add WeChat powcoder~~ that only the Student has
- Note that when `super` is used as above, it must be the first statement in the derived class's constructor

Overriding Methods and super Constructors

- Sometimes you want a derived class to have its own method, but that method includes everything the base class does
 - You can use the super reference in this case
- For example, here is class Person's method:
Add WeChat powcoder

```
WriteOutput ()  
  
public void writeOutput () {  
    System.out.println ("Name: " + name);  
}
```

Overriding Methods and super Constructors

- And here is Student's method:

```
public void writeOutput()  
{  
    System.out.println("Name: "+getName());  
    System.out.println("Student Number: "  
        + studentNumber);  
}
```

Overriding Methods and super Constructors

- Student's method can better be written using ~~Assignment Project Exam Help~~
~~super~~

```
public void writeOutput()  
    super.writeOutput();  
    Add WeChat powcoder  
    System.out.println("StudentNumber:"  
                      + studentNumber);  
}
```

Overriding Methods and super Constructors

- Note: Unlike the case when `super` is used in a constructor, inside a method `super` does not have to be used in the first statement
<https://powcoder.com>
- Note that you can form a new class from a derived class and can build inheritance to multiple levels
- Eg: class `Undergraduate` is derived from `Student` (see textbook Listing 8.4)

The final Modifier

- It is possible to specify that a method ~~Assignment Project Exam Help~~ [cannot be overridden](https://powcoder.com) in a sub-class by adding the **final** modifier to the method heading
[Add WeChat powcoder](https://powcoder.com)
- Eg:

```
public final void specialMethod ()  
{  
    // method body  
}
```

The final Modifier

- An entire class may be declared as `final`, which means it cannot be used as a base class to derive another class
- Eg: the Java API class `String` is declared as `final`

[Assignment Project Exam Help](https://powcoder.com)

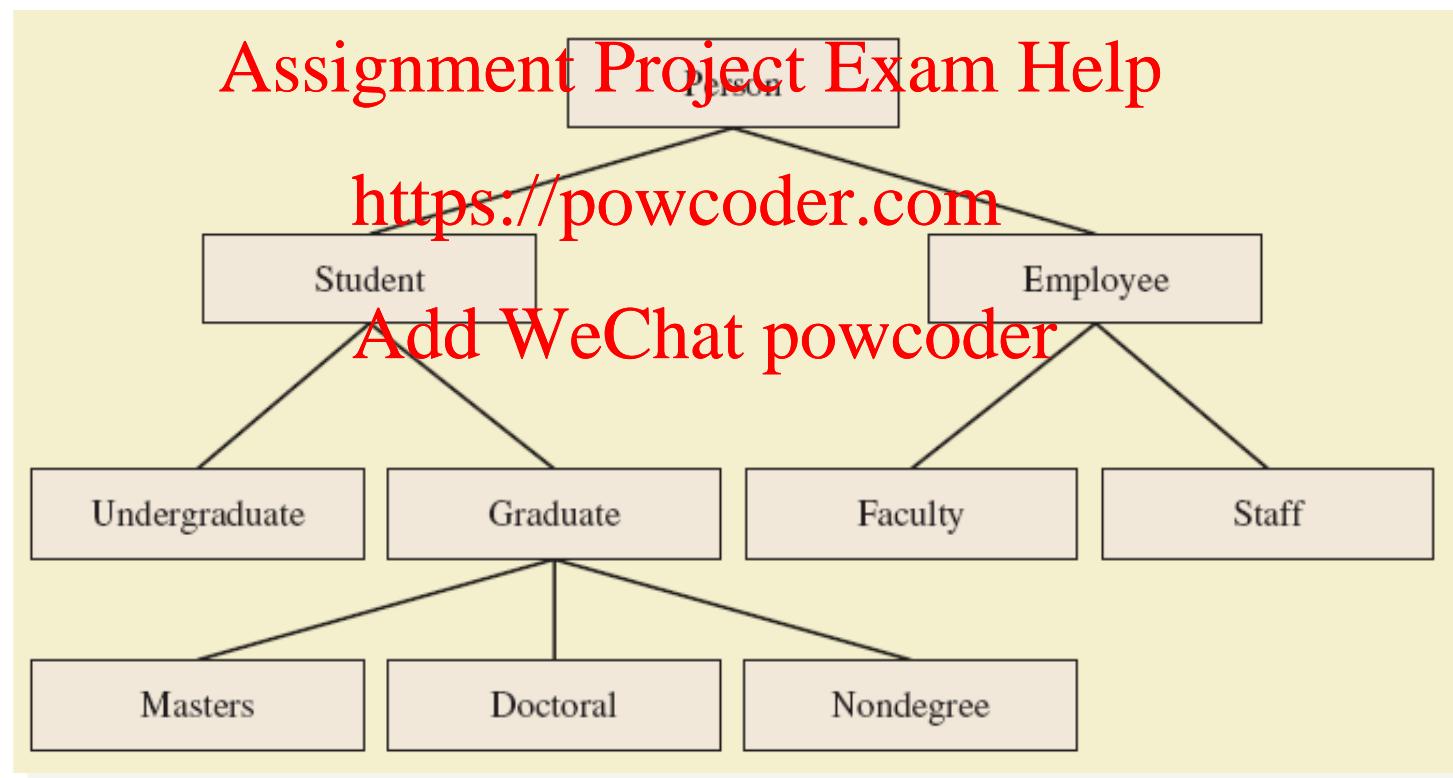
<https://powcoder.com>

[Add WeChat powcoder](https://powcoder.com)

```
public final class String extends Object  
{  
    . . . . .  
}
```

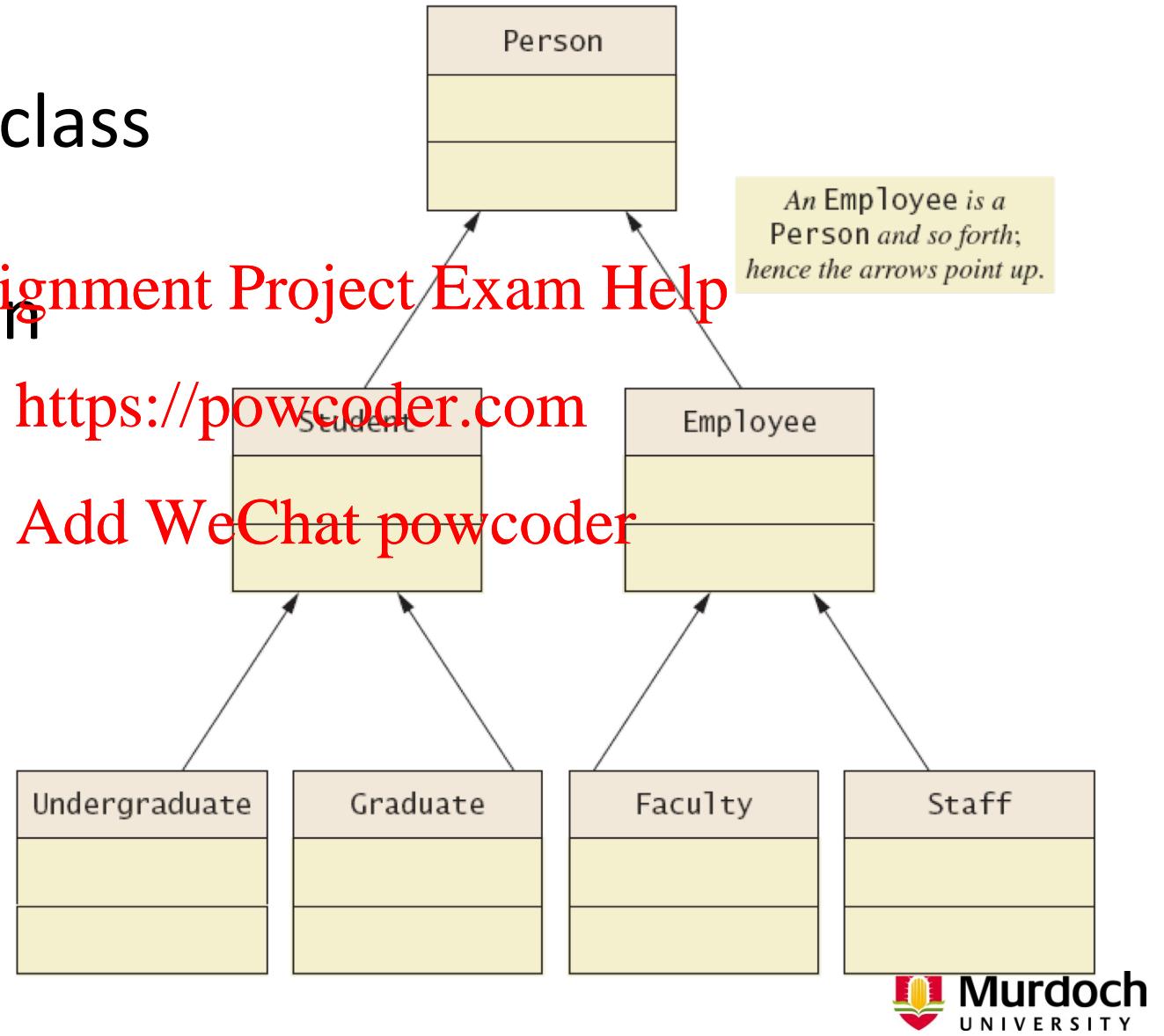
Class Hierarchies

Figure 8.1 A class hierarchy



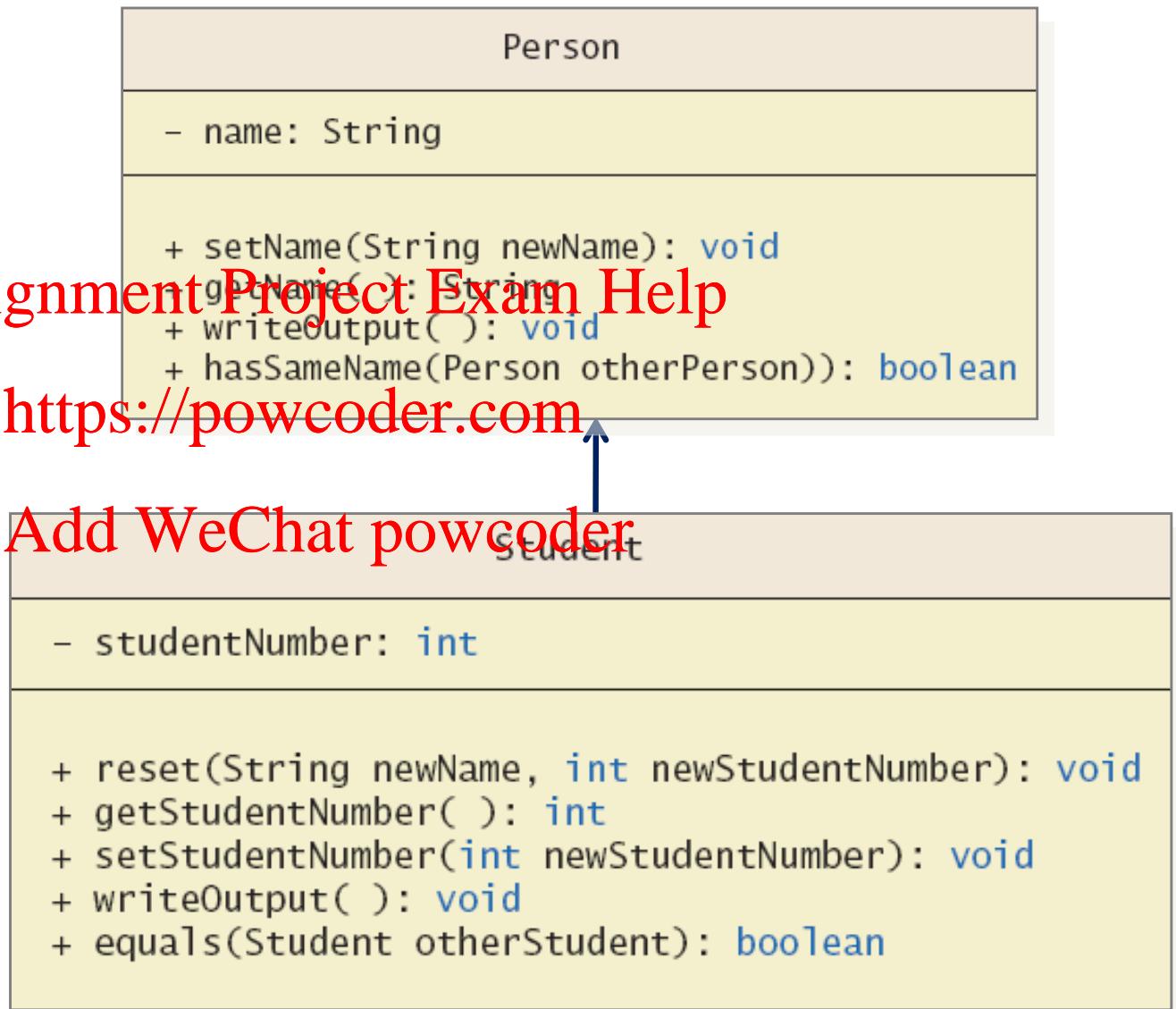
UML Inheritance Diagrams

- Figure 8.2 A class hierarchy in UML notation

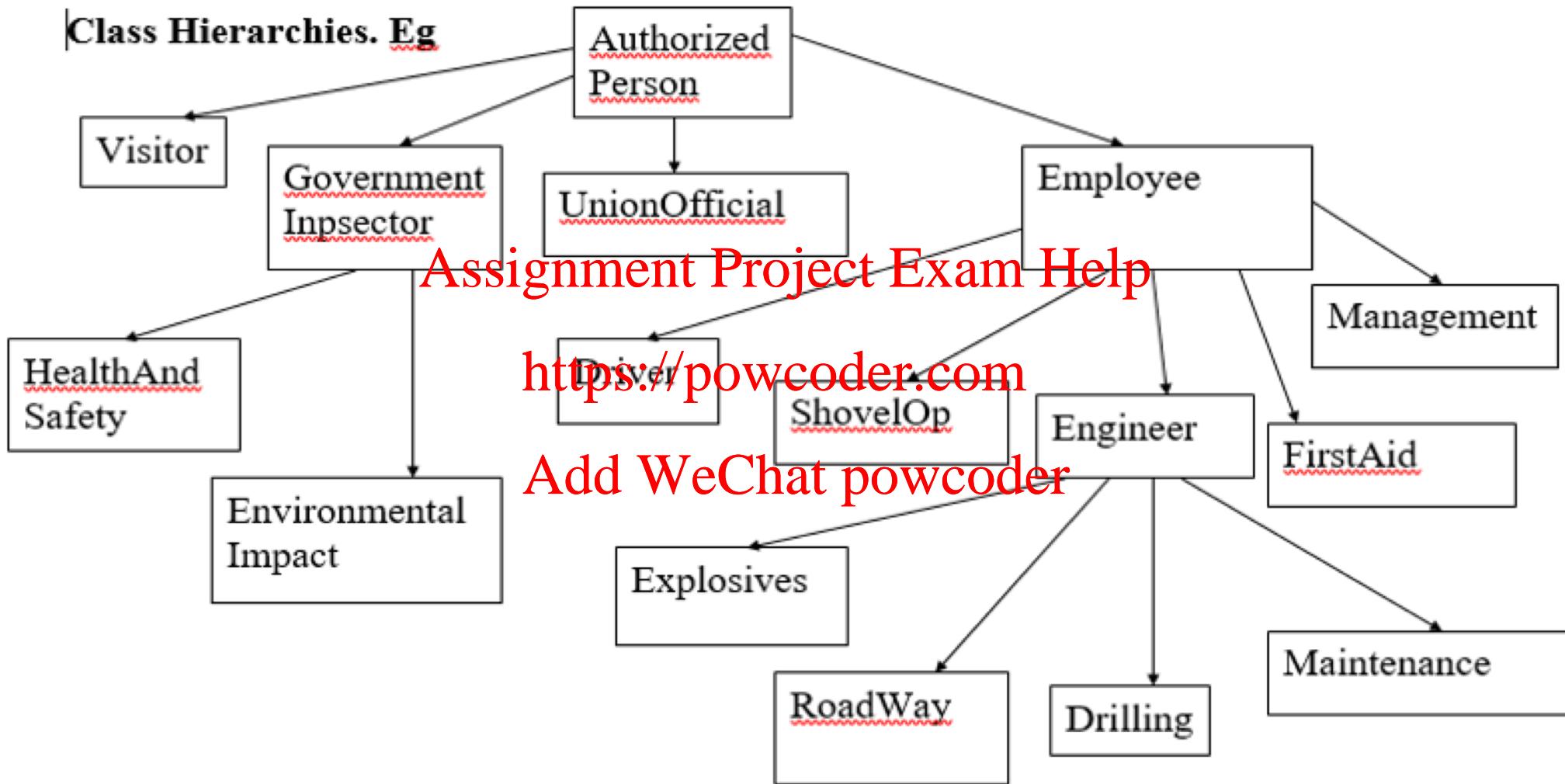


UML Inheritance Diagrams

- Figure 8.3
Some details
of UML class
hierarchy
from
figure 8.2



Class Hierarchies



Note that in UML diagrams, the arrows point upwards - to the parent class



Using Inheritance

- The diagram on the previous slide shows a *hierarchy* of classes
- In this class hierarchy, many methods and instance variables may be inherited downwards from <sup>Assignment Project Exam Help
<https://powcoder.com></sup> super class to derived class
- Java supports this easily
- In a hierarchy, each class has at most one base **class** (super or parent), but can have several derived (sub or child) classes

Using Inheritance

- Now in the above hierarchy, every driller or a first-aid person is also an employee
- Thus every object of the class Driller is also an object of the class Employee
- One of the most useful aspects of inheritance is that a derived class object can be used wherever a super class object can be

Using Inheritance

- Eg:

```
Date today = new Date();  
Employee emp = new Employee();  
emp = userChooseEmployee();  
System.out.println("You have chosen to retire  
the following employee");  
emp.writeName();  
System.out.println("Are you sure(yes/no)?");  
Scanner kb = new Scanner (System.in);  
String reply = kb.next();
```

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

Using Inheritance

```
if (reply.equals("yes")) {  
    emp.FinalizerRecords(today);  
} else {  
    https://powcoder.com  
    System.out.println("Request ignored.");  
}  
  
System.out.println("End of this request.");
```

- This will work no matter whether `emp` refers to a Driller or a FirstAid object etc.

Polymorphism

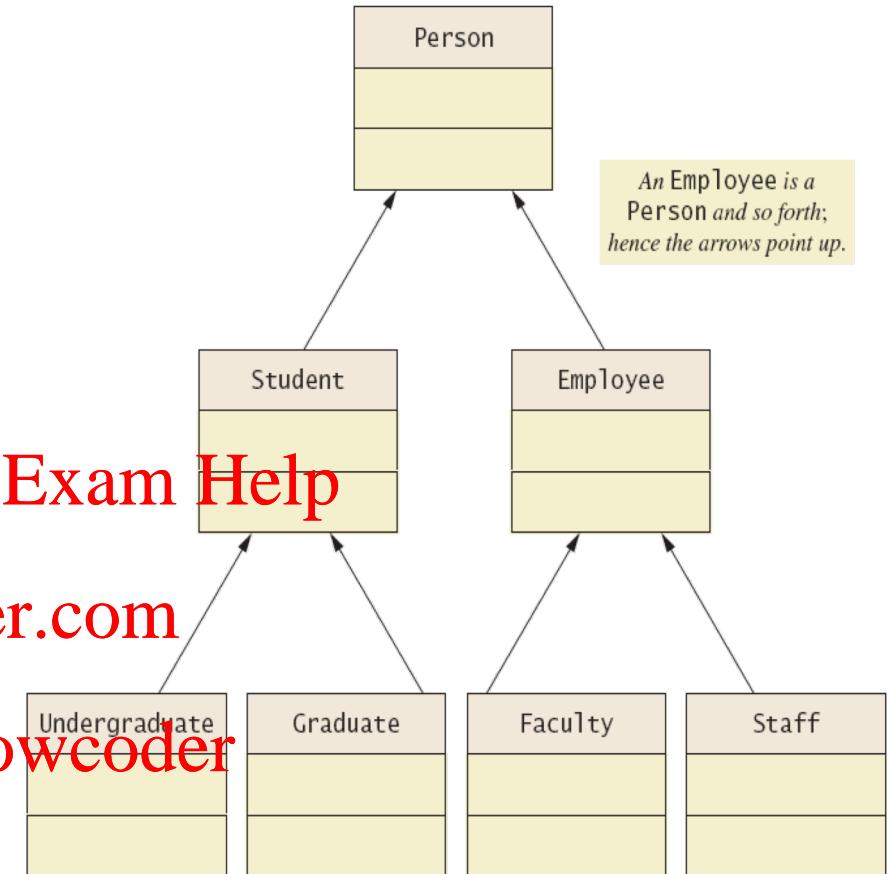
- Inheritance allows you to define a base class and derive classes from the base class
- **Polymorphism** allows you to make changes in the method definition for the derived classes and have those changes apply to methods written in the base class
- You will need to read more in the textbook under Chapter 8.3

Polymorphism

- Consider a program uses Person, Student, and Undergraduate classes

Assignment Project Exam Help

- E.g. if we want to set up a list of committee members (can be a person who are student or employee), it is better make an array of type **Person**
- Array of type **Person** can accommodate any class derived from it



Polymorphism

- Consider an array of **Person**

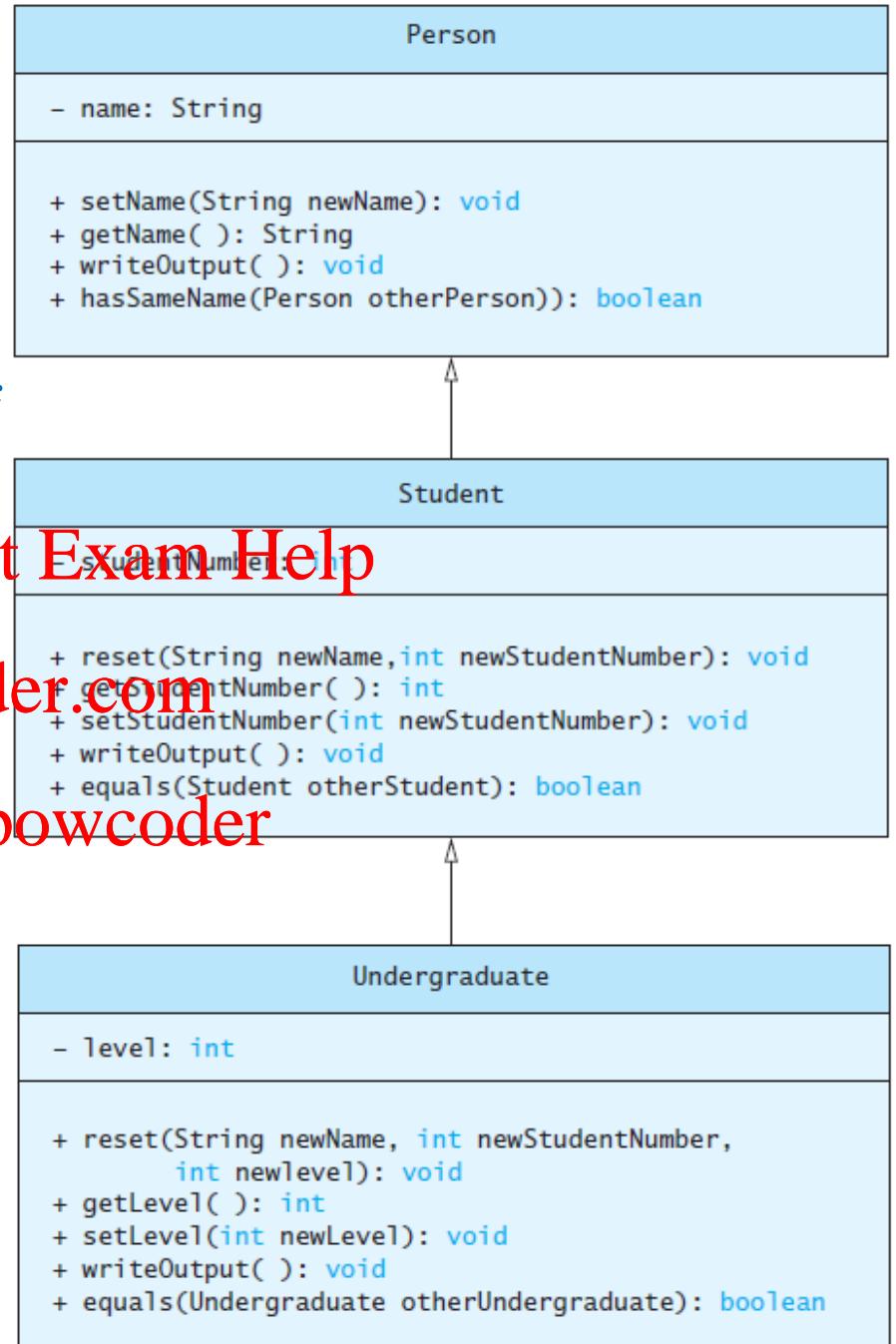
```
Person[] people = new Person[4];
```

- Since **Student** and **Undergraduate** are types of **Person**, we can assign them to **Person** variables

<https://powcoder.com>

```
people[0] = new  
Student("DeBanque, Robin",  
8812);
```

```
people[1] = new  
Undergraduate("Cotty, Manny",  
8812, 1);
```



Polymorphism

- Given:

```
Person[] people = new Person[4];  
people[0] = new Student("DeBanque, Robin",  
8812);
```

- When invoking <https://powcoder.com>

```
people[0].writeOutput();
```

Add WeChat powcoder
- Which `writeOutput()` is invoked, the one defined
for `Student` or the one defined for `Person`?
- Answer: The one defined for `Student`

An Inheritance as a Type

- The method can substitute one object for another
 - Called *polymorphism*
- This is made possible by mechanism
 - *Dynamic binding*
 - Also known as *late binding*

Dynamic Binding and Inheritance

- When an overridden method invoked
 - Action matches method defined in class used to create object using `new`
 - Not determined by type or variable naming the object
- Variable of any ancestor class can reference object of descendant class
 - Object always remembers which method actions to use for each method name

Polymorphism Example - listing 8.6

```
public class PolymorphismDemo
{
    public static void main(String[] args)
    {
        Person[] people = new Person[4];
        people[0] = new Undergraduate("Cotty, Manny", 4910, 1);
        people[1] = new Undergraduate("Kick, Anita", 9931, 2);
        people[2] = new Student("DeBanque, Robin", 8812);
        people[3] = new Undergraduate("Bugg, June", 9901, 4);

        for (Person p : people)
        {
            p.writeOutput();
            System.out.println();
        }
    }
}
```

Even though **p** is type Person, the **writeOutput** method associated with **Undergraduate** or **Student** is invoked depending upon which class was used to create the object

*Dynamic binding
Polymorphism*

Polymorphism Example

- Output

Assignment Project Exam Help

Name: Cotty, Manny

Student Number: 4010

Student Level: 1

Name: W.H. Manta

Student Number: 9931

Student Level: 2

Name: DeBanque, Robin

Student Number: 8812

Name: Bugg, June

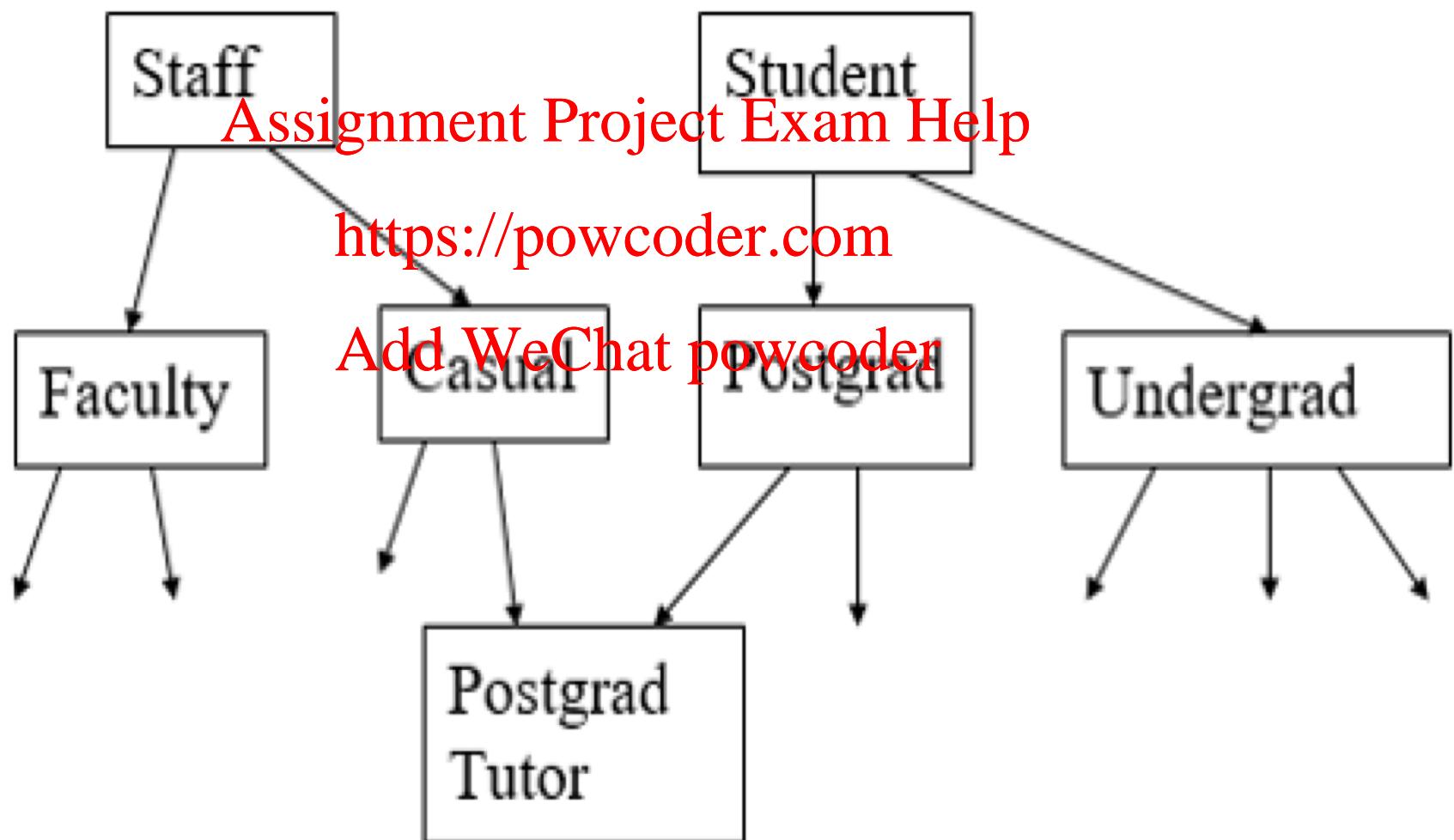
Student Number: 9901

Student Level: 4

Multiple Inheritance

- Occasionally, the natural description of a problem suggests a different form of inheritance, not like an upside down tree
 - For example, a postgraduate tutor may be both a staff member and a student
 - We may need methods to deal with paying them for taking lab classes, and methods for dealing with their student number, HECS fees and unit results
 - We want to inherit these methods from different super classes
- This is called **multiple inheritance**

Multiple Inheritance



Java Interfaces

- O-O languages get confused with multiple inheritance
Assignment Project Exam Help
 - Eg: if a postgrad tutor changes their office phone number, do we use the changeOfficePhone method supplied in the Staff class or in the Postgrad class?
- Some O-O languages provide ways to deal with this

Java Interfaces

- Java does **not** allow multiple inheritance except in a very special case:
 - One of the ~~super~~ classes must be an **interface**, which is like a class with methods with no bodies. (Do not confuse two uses of the word in this topic)
Add WeChat powcoder
- A Java **interface** is a collection of constants and method declarations
 - The method declarations do not include an implementation (i.e. there is no method body)

Java Interfaces

- A derived **class that** extends a base class can also implement an interface to gain some **Additional Project Exam Help**

- An interface <https://powcoder.com> has the following general form:

Add WeChat powcoder

```
// File: InterfaceName.java
public interface InterfaceName {
    constant definitions
    method declarations (without
                        implementations)
}
```

Java Interfaces

- A class definition then implements an interface as follows:

```
public class SomeClass extends  
    SomeParent implements  
InterfaceName  
{  
    Add WeChat powcoder  
    // body of the class SomeClass  
}
```

- You will know that an **interface** is involved if you see the word **implements** which is used instead of extends for interfaces

Java Interfaces

- Eg:

public class ButtonDemo extends JFrame
implements ActionListener

- Here we do not inherit any code from ActionListener except the ability to treat ButtonDemo objects as ActionListener Objects

The Swing Package and GUI Programming

- A *Graphical User Interface* (GUI) is a system of visible components (such as windows, menus, buttons, text fields) which allow a program to interact with a user
 - Modern <https://powcoder.com> windowing interfaces to allow the user to make choices with a mouse
 - **Swing** is a package that comes with Java 2, and contains classes for creating these sorts of components (graphics) and other classes which help them to be used (GUI programming)

The Swing Package and GUI Programming

- GUI programming, i.e. writing programs that set up and use GUIs, is complicated (but made much easier by the swing library)
<https://powcoder.com>
 - The swing library makes extensive use of inheritance
 - Swing can be viewed as an improved version of an older package called the **Abstract Windows Toolkit (AWT)**

The Swing Package and GUI Programming

- Also, designing and implementing a GUI using ~~Assignment Project Exam Help~~ requires skill at *event-driven programming*
<https://powcoder.com>
 - That is, a certain way of programming which makes use of objects representing events such as mouse click, keyboard press, windows becoming visible, etc.
 - GUI programming is advanced and we just give a brief overview here

The Swing Package and GUI Programming

- GUIs in Java are often managed by special programs called Applets which run in Internet browsers <https://powcoder.com>
 - Setting up an applet is easy (and you will see this in many other units)
 - Programming the working of an applet is much the same as programming a GUI application (i.e. a non-applet GUI program) and needs the same understanding of inheritance, the swing library and event-handling

Brief Overview of Java Event Handling

- What the user sees is determined by what visible ~~Assignment Project Exam Help~~ components the programmer "adds" to a frame (JFrame object).
<https://powcoder.com>
- In Java, a **frame** is a window that has a border, a place for a title, various buttons along the top border (eg: close button), and other built-in things
 - What we usually call a "window" Java calls a "frame"

Brief Overview of Java Event Handling

- The layout of the frame (window) is controlled by the programmer and a layout manager object <https://powcoder.com>
- The user interacts with the application by:
 - Clicking on a window's close button
 - Clicking on a button to choose one of the program's options
 - Making a choice from a menu
 - Entering text in a text field

Brief Overview of Java Event Handling

- When you perform an action (like mouse-clicking, keyboard presses) on a graphical component you generate an **event**
<https://powcoder.com>
- In **event-driven programming** the program responds to events
- The program responds to events that occur:
 - Whenever the user chooses, and
 - In whatever order the user chooses

Brief Overview of Java Event Handling

- *Events* are said to be **fired** by the component which they happen to
- The events ~~will not cause anything else to happen unless a *listener object* has been added to the firing component~~
~~<https://powcoder.com>~~
~~Add WeChat powcoder~~
 - Zero, one or more listening objects can be added
- Eg: in swing every object that can fire events, such as a button that might be clicked, can have one or more listener objects

Brief Overview of Java Event Handling

- If an event is fired then all the listening objects attached to the firing object are notified <https://powcoder.com>
- The listening object can have a method which says what to do if that particular event is fired
- This method is called an ***event handler***
- The programmer defines these event-handler methods

Brief Overview of Java Event Handling

- Most **swing** objects have methods for getting or setting their properties like what text is written on them or whether they are clickable
<https://powcoder.com>
 - Check on-line documentation for details
[Add WeChat powcoder](#)
 - Event handlers can change the component's properties, or remove or add components, or listeners, or do some calculation, or change the whole look of the GUI or close the whole program down

Brief Overview of Java Event Handling

- Thus a GUI program consists of three types of software:
 - Components that make up the Graphical User Interface
 - Listeners that receive the events and respond to them
 - Application code that does useful work for the user

Brief Overview of Java Event Handling

- A GUI program consists of a collection of graphical components that are all placed inside one or more windows - called <https://powcoder.com> container objects
- A frame (JFrame) object is a container object, so GUI components can be placed in it

Brief Overview of Java Event Handling

- Like all software objects, a frame-object is actually a section of main memory that holds information and methods
<https://powcoder.com>
- The Java system, with the help of the operating system and the graphics board, paints a picture on the computer monitor that represents the frame

GUI Components

- The GUI components are – windows, labels, text fields or text areas, buttons, etc.
- The components (labels, text areas/text fields and buttons) are added to the content pane (the area below the title bar and inside border) of a window and not to the window itself
- All GUI components are objects in Java and therefore are instances of a particular class type

GUI Components

- Below are some of the GUI components which can be created from ~~Assignment Project Exam Help~~ swing classes (contained in package `javax.swing`):
<https://powcoder.com>
 - JLabel – an area where un-editable text or ~~Add WeChat powcoder~~ icons can be displayed
 - JTextField – an area in which the user inputs data from the keyboard
 - It can also display information
 - It can have only one line of text

GUI Components

- JTextArea – an area as in JTextField above [Assignment Project Exam Help](#)
 - It can have many lines of text
<https://powcoder.com>
- JButton – an area that triggers an event when clicked [Add WeChat powcoder](#)
- JPanel – a container in which components can be placed and organized

Windows

- The GUI component **window** can be created using an instance of **JFrame** class
- The swing library class <https://powcoder.com> **JFrame** provides various methods to control attributes of a window
- The attributes associated with windows are:
 - Title
 - Width and height (in pixels)

Windows

- Some methods provided by the JFrame class:
 - `JFrame(String title)`
 - Constructor for creating a JFrame with a title
 - `Container getContentPane()`
 - Returns the content pane of the JFrame, which has the add method for adding components
 - `void setSize(int width, int height)`
 - Method to set the size of the window
 - Eg: `myWindow.setSize(500, 300);`
 - `void setTitle(String title)`
 - Method to set the title of the window

Windows

- Some methods provided by the JFrame class:
 - void setVisible(boolean b)
 - Method to display window in the program
 - Displays window on the screen if b is true
 - public void ~~Add Assignment Project Exam Help~~ addWindowListener(WindowEvent e)
 - Method to register a window listener object to a JFrame
 - public void ~~Add WeChat powcoder~~ setDefaultCloseOperation(int operation)
 - Method to determine action to be taken when the user clicks on window closing button, x, to close the window
 - Eg:

```
setDefaultCloseOperation(EXIT_ON_CLOSE)
;
```

Windows

- Some methods provided by the `JFrame` class
`Assignment Project Exam Help`
 - `void setBackground(Color c)`
<https://powcoder.com>
 - `void setForegroundColor(Color c)`
`Add WeChat powcoder`

Windows

- There are two ways to create a window in an application:
Assignment Project Exam Help
- The first way:
<https://powcoder.com>
 - Declare object of type `JFrame`
Add WeChat powcoder
 - Instantiate the object using `new`
 - Use various methods to manipulate window

Windows

- Alternatively:
 - Create the class containing the application program by extending definition of class <https://powcoder.com> JFrame using inheritance
 - The new class can use features such as methods it inherits from the existing class (JFrame), and can add some functionality of its own

Control Pane

- Content Pane is the inner area of GUI a window (below the title bar) and inside the border) <https://powcoder.com>
- GUI components are added to the content pane through a container class
- To access the content pane:
 - Declare reference variable of type Container
 - Use method getContentPane of class JFrame

Control Pane

- Eg:
Container c1 = getContentPane();
`c1 = getContentPane();
https://powcoder.com;`
- or,
`Add WeChat powcoder`
Container c1 = getContentPane();
- In order to design the layout to decide where to place the GUI components in the content pane, the class Container provides the method setLayout

Control Pane

- The components can be added/attached to the content pane by using method `add` of the `Container` class `Assignment Project Exam Help`
- The class <https://powcoder.com> is contained in the package `java.awt` `Add WeChat powcoder`
- To use this class in your program, you need to include either the statement:

```
import java.awt.*;
```
- or

```
import java.awt.Container;
```

Labels

- A label is a special kind of text that can be added to a `JFrame` (or to any of a number of other [Assignment](#) [Projects](#) [Exam Help](#))
- It provides [instruction](https://powcoder.com) or information on a GUI
[Add WeChat powcoder](#)
- It displays a single line of read-only text, an image or a mixture of both
- Labels are created by instantiating objects of class `JLabel` (which is contained in the package `java.swing`)

Labels

- Eg: Give a string as an argument to the constructor for the `JLabel` class:

```
JLabel label1;  
label1 = new JLabel("Please don't  
click that button");  
c1.add(label1, BorderLayout.CENTER);
```

- Eg: set string describing `label2` as right-justified

```
JLabel label2;  
label2 = new JLabel("Enter your  
name:",  
SwingConstants.RIGHT);
```

Text Fields and Text Areas

- Text fields (objects of class `JTextField`) are single-line areas in which the user can enter text (via keyboard) or the program can display text
<https://powcoder.com>
- When the user enters data into a text field and presses the Enter key, an action event (ActionEvent) occurs
[Add WeChat powcoder](#)
- If the program has registered an event listener, the listener will process the event enabling the program to use the data entered in the text field

Text Fields and Text Areas

```
JTextField mytext;  
mytext = new JTextField(50);
```

- This statement instantiates the object <https://powcoder.com> and sets the width of this text field to 50 characters
- The object mytext will be added to the content pane using the add method of Container class

Text Fields and Text Areas

- Some methods provided by JTextField class:
 - public JTextField(int size)
 - Constructor to set the size of the text field
 - public JTextField(String str)
 - Constructor to initialise object with text specified by str <https://powcoder.com>
 - public void setText(String str)
 - Method to set text of text field to string specified by str [Add WeChat powcoder](#)
 - public String getText()
 - Method to return the text contained in the text field
 - public void addActionListener(ActionListener e)
 - Method to register a listener object to a JTextField

Text Fields and Text Areas

- **Text areas** (objects of class `JTextArea`) are areas in which many lines of text can be entered and/or displayed
<https://powcoder.com>
- Eg:
 Add WeChat powcoder

```
JTextArea text2 = new JTextArea(10,  
50);
```
- `text2` is big enough to hold 10 lines, where each line can hold up to 50 characters

Text Fields and Text Areas

- Similar methods, as those in JTextField class, ~~Assignment Project Exam Help~~ are also available in JTextArea class
- Eg: getText(), setText(String str),
addActionListener(ActionListener e)
Add WeChat powcoder
- The text fields/areas are then added to the content pane of a window using the add method and a layout manager

Buttons

- A button (created with class JButton) is a component the user clicks to trigger an action (ActionEvent). The text on the face of a button is called button label
<https://powcoder.com>
Add WeChat powcoder
- An ActionEvent can be processed by any ActionListener object
- To create a button, we use the same techniques as creating labels and text fields

Buttons

- Some methods provided by the class

JButton:

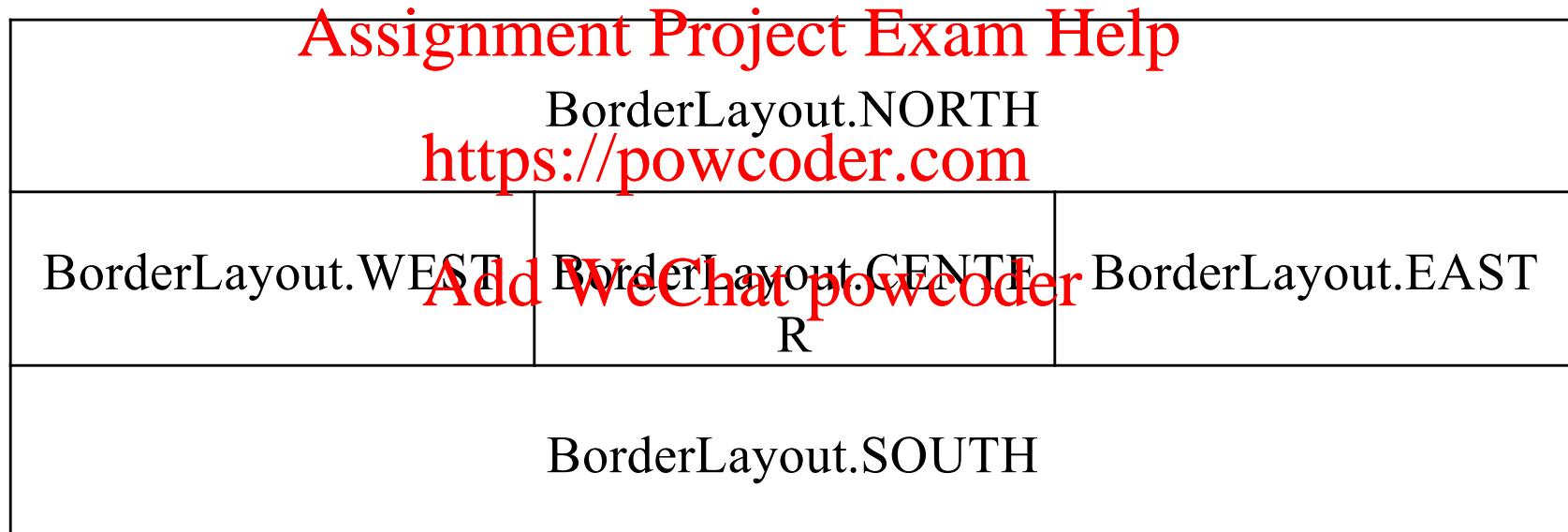
- `public JButton(String str)`
 - Constructor to initialise the object to text specified by str
- `public void setText(String str)`
 - method to set text of the button to string specified by str
- `public String getText()`
 - method to return the text contained in button
- `public void addActionListener(ActionListener e)`
 - method to register a listener object to the button object

Layout Managers

- Layout Managers are objects that decides how components will be arranged in a container
- Assignment Project Exam Help
- Some types of layout managers:
 - BorderLayout
 - FlowLayout
 - GridLayout
- Each type of layout manager has rules about how to rearrange components when the size or shape of the container changes

The Border Layout Manager

- It has five regions that can each have one component added to them:



```
c1.setLayout(new BorderLayout());
```

```
...
```

```
c1.add(label1, BorderLayout.NORTH);
```

The Flow Layout Manager

- Flow is the simplest layout manager; it displays's components from left to right in the order they are added to the container
- The **add** method has one parameter which is the component to add
<https://powcoder.com>

```
Container c2 = getContentPane();
Add WeChat powcoder
c2.setLayout(new FlowLayout());
JLabel label1=new JLabel("1st label here");
c2.add(label1);
JLabel label2=new JLabel("2nd label there");
c2.add(label2);
```

The Grid Layout Manager

- The programmer specifies the number of rows and columns in the grid
- All regions in the grid are of equal size
- When the container changes size, each region grows or shrinks by the same amount

The Grid Layout Manager

- The following example creates a grid layout with two rows and three columns:

```
Container c3 = getContentPane();  
https://powcoder.com  
c3.setLayout(new GridLayout(2, 3));  
Add WeChat powcoder  
• • •  
c3.add(label1);  
c3.add(label2);
```

- Note that the rows are filled before columns in the grid

Handling an Event

- When button (`JButton`) is clicked, an event is created – called **action event**
- Action event sends a signal to another object, known as **action listener**
<https://powcoder.com>
- When the listener receives the message, it performs some action
[Add WeChat powcoder](#)
- Sending a message or an event to a listener simply means that some method (eg, `actionPerformed`) in the listener object is invoked with the event as the argument

Handling an Event

- This invocation happens automatically – there is no code corresponding to the method call
Assignment Project Exam Help
- However, you must specify two things:
 - For each JButton, you must specify a corresponding ~~Add WeChat powcoder~~ object – called **registering** the listener
 - You must define the methods that will be called when the event is fired (i.e., sent to the listener)
- Java does not allow us to instantiate an object of type ActionListener

Class ActionListener

- The class `ActionListener` (part of the package `java.awt.event`) handles action events
- It is a special type of class called an **interface** and contains the method `actionPerformed`
- An interface is a class that only contains the method headings (terminated with a semicolon) and not their definitions/implementations
- Java does not allow us to instantiate an object of type `ActionListener`

Class ActionListener

- One way to register an event is to create a class on top of ActionListener so that the required object can be instantiated
- Eg: Assignment Project Exam Help

```
private class MyButtonHandler implements  
    https://powcoder.com ActionListener {  
    public void actionPerformed  
        Add WeChat powcoder (ActionEvent e) {  
        //Code for tasks to be performed go here  
    } // end actionPerformed  
} // end class MyButtonHandler
```

Example: SimpleApp

```
//SimplApp.java - a simple example of a GUI program
//You should be able to give a brief description of what
//such a program will do and the steps involved
import javax.swing.*; //for JFrame, JButton, JLabel
import java.awt.*; //for Container, BorderLayout
import java.awt.event.*; //for WindowAdapter,
//WindowListener, ActionEvent
public class SimplApp extends JFrame {
    // define window's width and height in pixels
    private static final int WIDTH = 400;
    private static final int HEIGHT = 200;
```

Example: SimpleApp

```
// used for displaying text in the window
private JLabel infoLabel;
Assignment Project Exam Help
private class ButtonAction implements
    https://powcoder.com    ActionListener {
    public void actionPerformed(ActionEvent e) {
        Add WeChat powcoder
        infoLabel.setText("You fool !!!");
    } //end of actionPerformed
} //end of class ButtonAction
```

Example: SimpleApp

```
// used to destroy/close the window  
private class WindowDestroyer extends  
Assignment Project Exam Help  
WindowAdapter {  
    public void windowClosing(WindowEvent e) {  
        dispose();  
        Add WeChat powcoder  
        System.exit(0);  
    } //end of windowClosing()  
} //end of class WindowDestroyer
```

Example: SimpleApp

```
// Below is the constructor for the class SimplApp
public SimplApp(String windowTitle) {
    super(windowTitle);
    setSize(WIDTH, HEIGHT);
    // create content pane to add components to
    window
    Container c1 = getContentPane();
    Add WeChat powcoder
    c1.setLayout( new BorderLayout());
    // create a label component with the String
    centred
    infoLabel = new JLabel( "Initial",
                           JLabel.CENTER);
    c1.add( infoLabel, BorderLayout.CENTER);
```

Example: SimpleApp

```
// create a button component
JButton button1=new JButton("Don't Press
Me!");
c1.add(button1,BorderLayout.NORTH);
//goes at top
// add an action event to button
ButtonAction myAction = new ButtonAction();
button1.addActionListener(myAction);
// add action event to window close button
WindowDestroyer myListner = new
WindowDestroyer();
addWindowListener( myListner);
} //end of SimplApp constructor
```

Example: SimpleApp

```
public static void main(String[] args) {  
    // calls constructor  
    SimplApp app = new SimplApp("Zzzz");  
    // display window on the screen  
    app.setVisible(true);  
    System.out.println("Finished  
                        SimplApp.main()");  
} //end of SimplApp.main()  
} //end of SimplApp class
```

Example: BinarySearch

```
// BinarySearch.java revised (by P S Dhillon) from  
// Deitel and Deitel  
// Binary search of an array  
import java.awt.*;  
import java.awt.event.*;  
import javax.swing.*;  
import java.text.*;  
public class BinarySearch extends JFrame  
    implements ActionListener {  
    JLabel enterLabel, resultLabel;  
    JTextField enterField, resultField;
```

Example: BinarySearch

```
JTextArea initial, output;  
int arr[];  
String display = "";  
public static void main (String [] args) {  
    BinarySearch myApplication = new  
        BinarySearch ("Binary Search");  
    myApplication.setVisible (true);  
    myApplication.setDefaultCloseOperation (  
        EXIT_ON_CLOSE);  
} // end main
```

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

Example: BinarySearch

```
// constructor for BinarySearch
public BinarySearch(String title) {
    super (title);
    setSize(800, 300);
    Container c = getContentPane();
    c.setLayout( new FlowLayout() );
    // set up JLabel and JTextField for user input
    enterLabel = new JLabel( "Enter an integer
                           key to search" );
    c.add( enterLabel );
```

Example: BinarySearch

```
enterField = new JTextField( 5 );
enterField.addActionListener( this );
c.add( enterField );
// set up JLabel and JTextField for displaying
results
resultLabel = new JLabel( "Result" );
c.add( resultLabel );
resultField = new JTextField(20);
resultField.setEditable( false );
c.add( resultField );
```

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

Example: BinarySearch

```
// create array and fill with odd integers 1 to 29
arr = new int[ 15 ];
for (int i = 0;i < arr.length;i++)
    arr[ i ] = 2 * i + 1;
// set up JTextArea for displaying the array
contents Add WeChat powcoder
JTextArea initial = new JTextArea(3, 60 );
c.add( initial );
```

Example: BinarySearch

```
// build the initial array for displaying  
String arrayContents="Contents of array:\n";  
for (int i = 0;i < arr.length;i++)  
    arrayContents = arrayContents + (arr[i])  
                + "    " ;  
  
initial.SetText(arrayContents);  
  
// set up JTextArea for displaying comparison  
output = new JTextArea(10, 60);  
c.add( output );  
}  
} //end of constructor
```

Assignment Project Exam Help
<https://powcoder.com>

Example: BinarySearch

```
// obtain user input and call method binSearch
public void actionPerformed(ActionEvent e) {
    String searchKey = e.getActionCommand();
// initialize display string for the new search
    display = "Portions of array searched:\n";
    Add WeChat powcoder
// perform the binary search
    int index = binarySearch(arr,
    Integer.parseInt(searchKey));
    output.setText( display );
```

Example: BinarySearch

```
if ( index != -1 )  
    resultField.setText("Value found at  
                        index " + index);  
else    https://powcoder.com  
        resultField.setText("Value not found");  
} //end of actionPerformed method
```

Assignment Project Exam Help
Add WeChat powcoder

Example: BinarySearch

```
public int binarySearch(int ar[], int key) {  
    int first = 0; Assignment Project Exam Help  
    int last = ar.length - 1;  
    int mid; https://powcoder.com  
    while (first <= last) Add WeChat powcoder  
        mid = (first + last)/2;  
    // The following line is used to display the part  
    // of the array currently being manipulated  
    // during each iteration of the binary search loop.  
    buildOutput( first, mid, last );
```

Example: BinarySearch

```
if (key == ar[mid]) // match found
    return mid; // exit
else if(key < ar[mid])
// search low end of array
    last = mid - 1;
else //search high end of array
    first = mid + 1;
} // end while
return -1; // match not found
} // end binarySearch
```

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

Example: BinarySearch

```
// Build one row of output showing the current  
// part of the array being processed.  
Assignment Project Exam Help  
void buildOutput(int low,int mid,int high) {  
    DecimalFormat twoDigits = new  
        DecimalFormat( "00" );  
Add WeChat powcoder  
    for (int i = 0;i < arr.length;i++) {
```

Example: BinarySearch

```
if ( i < low || i > high )  
    display = display + ":";  
else if ( i == mid ) //mark middle element  
    display = display + twoDigits.format( arr[ i ] ) + "* ";  
else Add WeChat powcoder  
    display = display + twoDigits.format( arr[ i ] ) + " " ;  
} // end for  
display = display + "\n";  
}// end of buildOutput  
}//end of class
```



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder
End of Topic 7