ICT167 ANS7

Inheritance*:

- Allows us to define a new class based on a (general class) that already exist
- The new class similar to existing class. Can use all facilities of the existing class + some new characteristic

- Done through keyword: extend

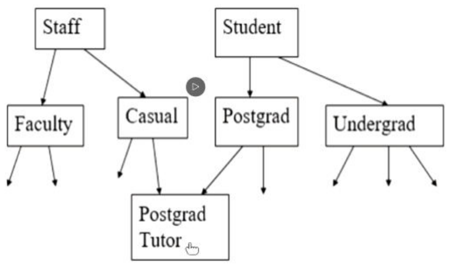
Base class (super class/parent class): this class that is used as the basis for defining a new class

Derived class (sub-class/child class): this class is based on the base class

Thus, the derived class INHERITS from the base class. It is *EXTENDED*

Single Inheritance: child class INHERITS characteristics from one parent

Multiple inheritance:



- Child class INHERITS characteristics from MORE than one parent
- Child class to inherit methods from different super classes

- Avoid using multiple inheritance with JAVA as it gets confused as to which method to invoke. Java doesn't allow except
- One of superclass must be an: Java interface

Java Interface: (allow multiple inheritance)

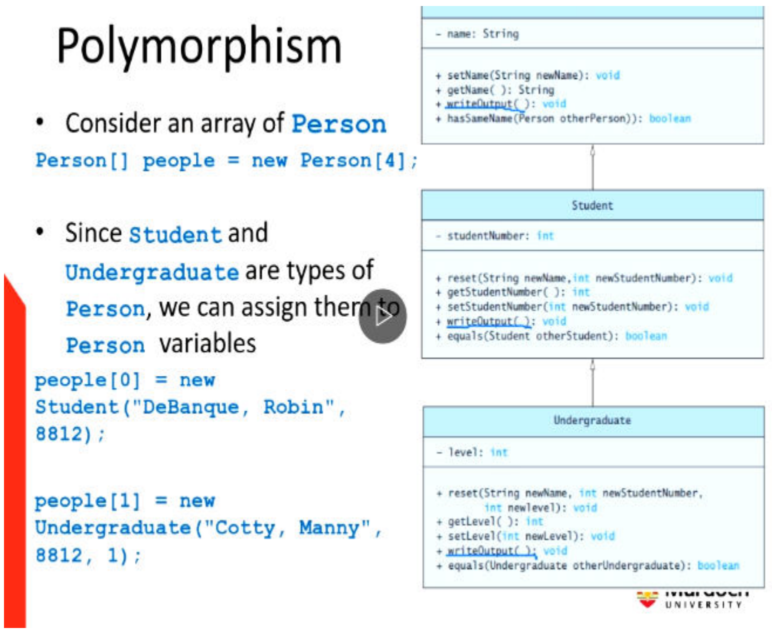
- Collection of constants + method declaration (without body) that allow multiple inheritance with Java

- Super class with methods without no body/implementations [INSERT CODE]
- Then child class must extend it [INSERT CODE]

Polymorphism*:

- Allows you to make changes in the method definition for the derived classes and those changes apply to methods in the base class
- Can substitute one object for another

- Knows level of inheritance we are dealing with. When we invoke a method (override method) Links method to the right derived class (clarify) ← Made possible by dynamic binding



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

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

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

- name: String
+ setName(String newName): void
+ getName( ): String
+ writeOutput( ): void
+ hasSameName(Person otherPerson)): boolean
Student
- studentNumber: int
+ reset(String newName, int newStudentNumber): void
+ getStudentNumber( ): int
+ setStudentNumber(int newStudentNumber): void
+ writeOutput( ): void
+ equals(Student otherStudent): boolean
Undergraduate
- level: int
+ reset(String newName, int newStudentNumber, int newlevel): void
+ getLevel( ): int
+ setLevel(int newLevel): void
+ writeOutput( ): void
+ equals(Undergraduate otherUndergraduate): boolean
UNIVERSITY

Dynamic binding:

- When an overridden method is invoked. The action matches method defined in class used to create object using new
- Variable of any ancestor class (super class/parent class) can reference object of descendant class
    - Object always remembers which method actions (belonging to different classes) to use for each method name

---

Overriding methods: (Overriding is possible because of Dynamic binding)

- When methods appear in both the base class and the derived class they are redefined
- The derived class overrides the base class since we assume the derived class we deal with more instance variables or more up to date instance variables (think base instance variables inherited + the instance variables)

- The method contains same name, same number, order and types of parameters
- Use super method to get/call to base method to get Instance variables not in the derived class
    - super.writeOutput() (refers/calls to super/base method)

Final modifier:

- Final is put in front of method to STOP method from being overridden
- Final is put in front of class to STOP others inheriting base class or allowing the class to be a base class

- Useful for base class methods and not overriding them

GUI:

- GUI is a system of visible components that allow a program to interact with a user. Managed by applets which is special programs run on internet browser
- Components include-
    - Components for GUI
    - Listeners that receive the events and respond to them
    - Application code that does the useful work for user
- Considered an improved Abstract Windows Toolkit (AWT). Based on event-driven programming

- Use Swing package for Java which contains classes for these components
- GUI components are objects in JAVA thus must be instances of a class type

Java (Swing) event driven programming:

- A window is called a frame (JFrame object) with swing components on it (Content Pane) in Java. This is what user sees
- **Event or action event** is an action on GUI. When Events that are fired all listener objects are notified
- **Listener object or action listener-** when some event is fired this object may capture the event and perform an event handler method
- **Event handler method-** defined by the programmer says what to do

Methods to create window (JFrame object) in application:

1. Declare object of type JFrame and Instantiate the object using new
2. Create class containing the application by extending definition of class JFRAME using inheritance (refer to multiple inheritance)

Method to create content pane in application: [Must add]

- Refer to JavaAPI.docx but it is inner area of GUI (Below title bar and inside border.So inside window)
    - GUI components are added to content pane
- Container c1 = getContentPane();

Layout management:

- How objects will be arranges in a container.
- Three types of layout managers include-
    - BorderLayout
    - FlowLayout:

        - Displays components from left to right in the order they are added to the container
    - GridLayout

JFRAME/WINDOW (class) methods:

```
import javax.swing OR java.awt.* (awt was the bases for swing);
```

CONTAINER (class) methods:

```
Import java.awt.container OR java.awt.*;
```

JLABEL (class) methods:

```
import javax.swing
```

JBUTTON (class) methods:

```
import javax.swing
```

JTEXTFIELD (class) methods:

```
import javax.swing
```

JTEXTAREA (class) methods:

```
import javax.swing
```

ACTIONLISTNER (class) methods:

```
import java.awt.event OR java.awt.*;
```

```
Don’t instantiate an object of type: actionListener
```

INTERFACE (class) methods: (special class containing one method. No instantiate)

```
Import java.awt.*;
```