OBJECT ORIENTED PROGRAMMING

Inheritance

by
Dr. Nor Saradatul Akmar Zulkifli
Faculty of Computer Systems & Software Engineering
saradatulakmar@ump.edu.my

Content Overview

- What is Inheritance?
- Types of Inheritance
- Inheritance represented in a class diagram
- Superclass (Parent) and Subclass (Childs)

Learning Objectives

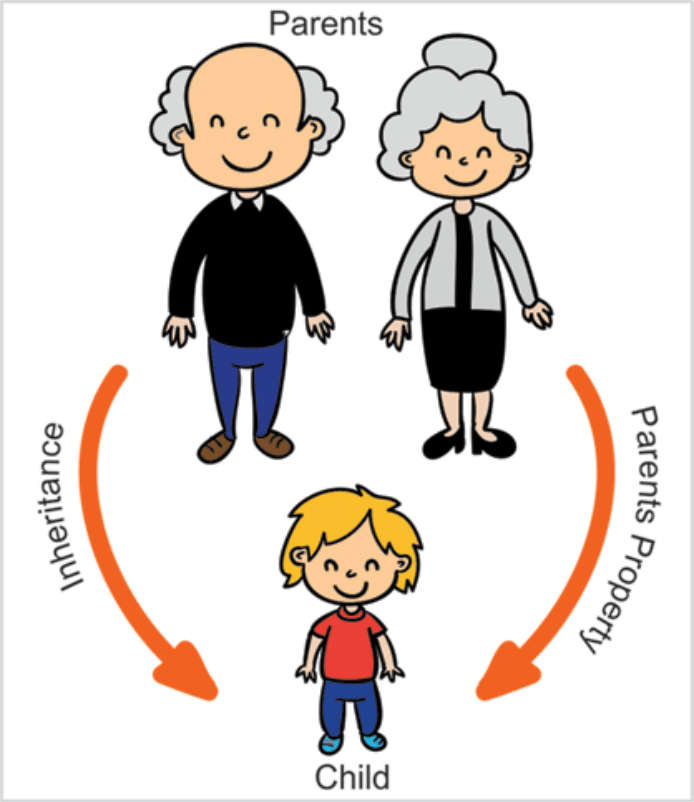
- To understand the basic concept of Inheritance
- To differentiate between inheritance types
- To define a child class in Java using `extends` keyword

REVIEW : INHERITANCE

- It enables objects to **inherit attributes and behaviors** from other objects
- Classes with properties in common can be grouped so that their common properties are only defined once
- **Superclass Vs. Subclass**

ADVANTAGE

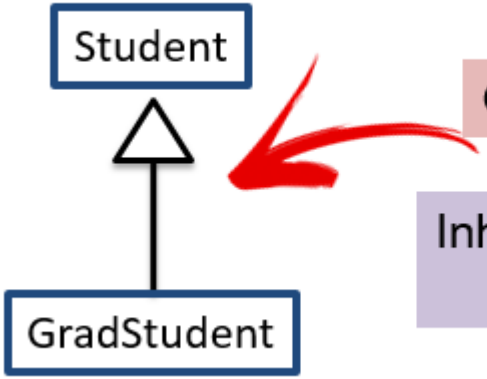
- Reduce the amount of new code that must be designed, written and tested each time a new program is develop



WHY?

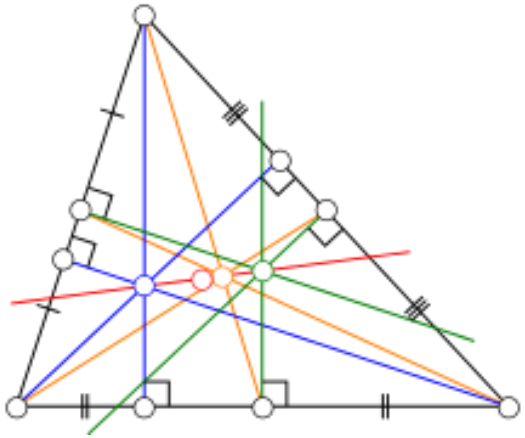
- The programmer can add new variables or methods to adjust a derived class or can even modify (override) the inherited ones

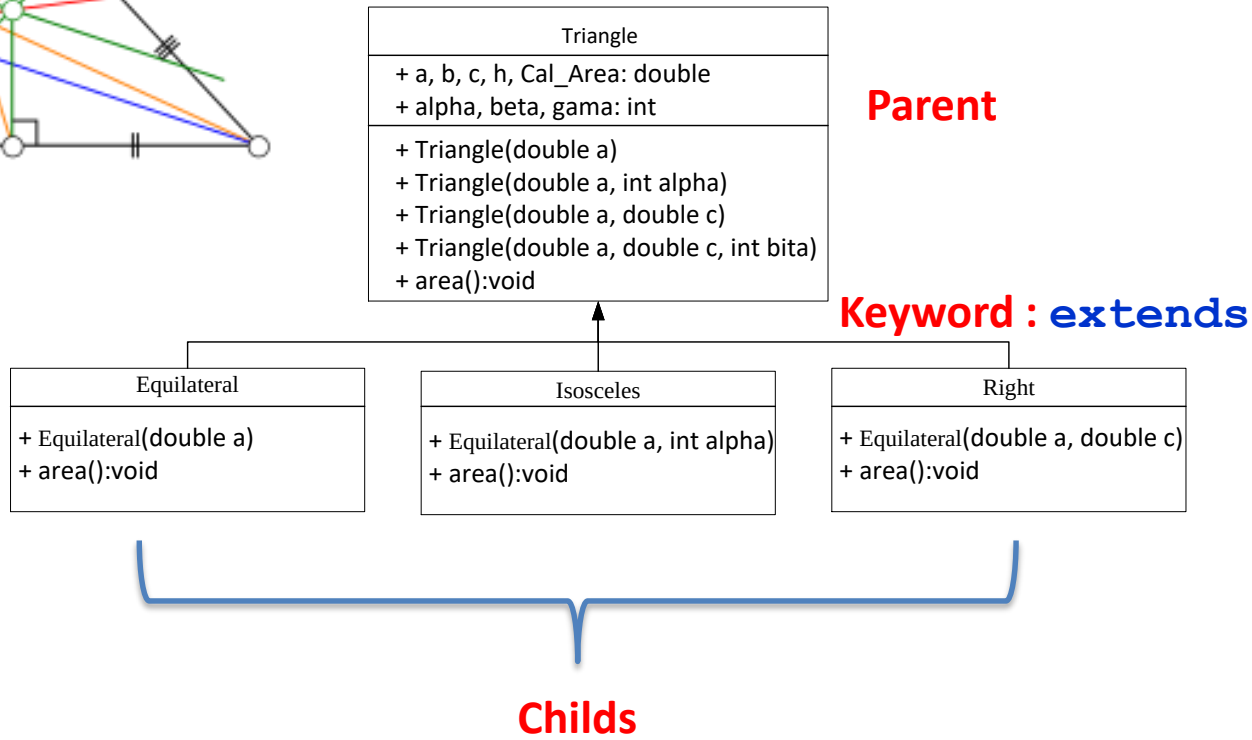
- Reduce the effort to design, implementation and testing new software by reusing the existing software components to create new ones.



Graphically shown the inheritance relationship

Inheritance creates an *is-a* relationship;
Child *is a* more specific version of the parents





Example:

```
class Equilateral extends Triangle
class Isosceles extends Triangle
class Right extends Triangle
```

TYPES OF INHERITANCE

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

Single

- A child class can be derived from a single/one parent class only
- Supports by JAVA ✓
- A parent class cannot access attributes and behavior of a child class.

Multiple

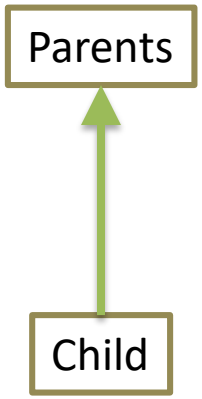
- A child class can be derived from more than one parent class
- Not support in JAVA ✗

INHERITANCE WITH JAVA

How to define a class is a child of a parent class?

1. Use the Java `extends` keyword

2. A child `extends` from a parents because …..
   it inherits properties from the parent and can add more new properties of its own.



THE extends KEYWORD

Syntax:

```
Class <ChildClassName>  extends  <ParentClassName>
{
      // data & methods specific to child
}
```

Example:

```
Class GradStudents  extends  Students
{
      // data & methods specific to child
}
```

HOW?

1. Define the Parent Class : Student

Student
- name : String - age : int
+ setName (String) : void + setAge (int) : void + getName ( ) : String + getAge ( ) : int

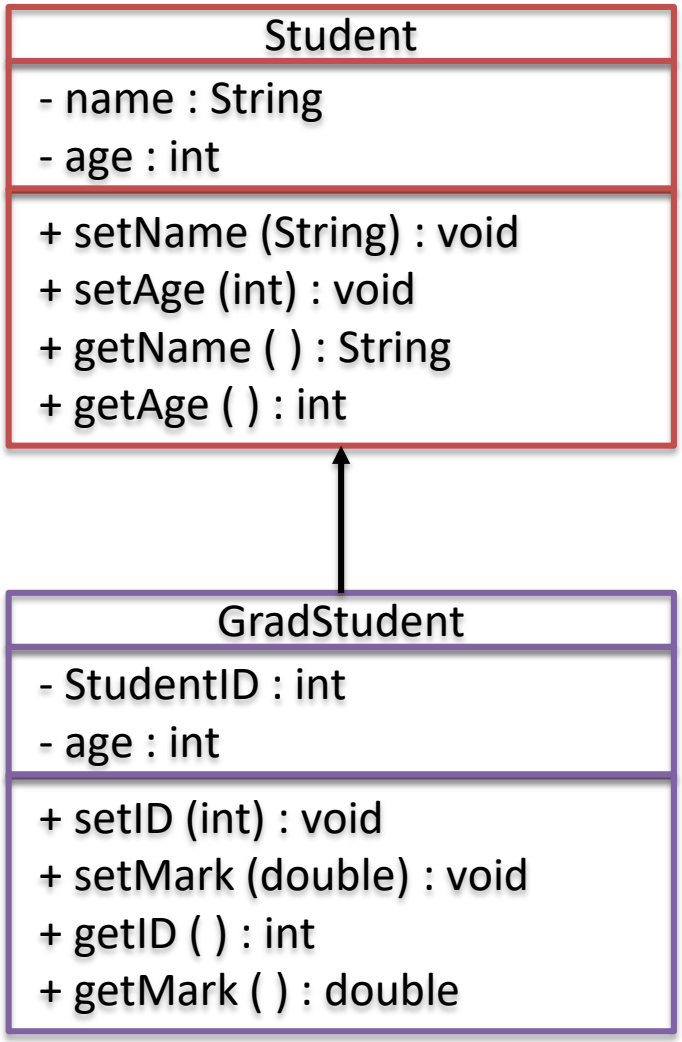
```
public class Student {

        private String name;
        private int age;
        public void setName (String name)
        {
                this.name = name;
        }
        public void setAge (int age) {
                this.age = age;
        }
        public String getName ( ) {
                return name;
        }
        public int getAge ( ) {
                return age;
        }
```

HOW?

2. Define the child Class : GradStudent

```
public class GradStudent extends Student
 {
      private int studentID;
      private double mark;
      public void setID (int matric)
      {
             this.studentID = matric;
      }
      public void setMark (double mark) {
             this.mark = mark;
      }
      public int getID ( ) {
             return studentID;
      }
      public int getMark ( ) {
             return mark;
      }
```



HOW?

3. Write a test program (or the main)

```
public class TestProgram {
    public static void main (String [ ] args) {
        Student s1 = new Student ( );
        GradStudent g1 = new GradStudent ( );
        s1.setName ("Ammar");
        s1.setAge (28);
        g1.setName ("Amira");
        g1.setAge (19);
        g1.setID (01889);
        g1.setMark (95);

        System.out.println ("Student name is " + s1.getName());
        System.out.println ("Student age is " + s1.getAge());
        System.out.println ("GradStudent name is " + g1.setName());
        System.out.println ("GradStudent age is " + g1.getAge());
        System.out.println ("GradStudent ID is " + g1.getID());
        System.out.println ("GradStudent mark is " + g1.getMark());
    }
```

HOW?

Output

```
run :
Student name is Ammar
Student age is 28
GradStudent name is Amira
GradStudent age is 19
GradStudent ID is 01889
GradStudent mark is 96.0
BUILD SUCCESSFUL (total time: 1 second)
```

INHERITANCE : Constructor

- Used `super` keyword to refer to the parent class and often used to invoke the parent's constructor
- A child's constructor is responsible for calling the parent's constructor
- To call the parent's constructor – the first line of child's constructor can be the `super` keyword.
- The `super` keyword can also be used to reference other variables and method defined in the parent's class

```
super ( );
```

Used to call the constructor from superclass (parents) with appropriate arguments

INHERITANCE : Overriding Methods

- A child class can override the definition of an inherited method in favor of its own.
- The new method must have the SAME SIGNATURE (name and parameters) as the parent's method BUT can have a DIFFERENT BODY (implementation
- The type of the object executing the method determines which version of the method is invoked

```
super ( );
```

- Invoked explicitly the parents method using `super` reference.
- Method with `final` modifier, cannot be overridden.
- *Shadowing variables* is when an overriding concept applied to data and should be avoided – cause unnecessarily confusing code.

INHERITANCE : Overriding Methods

Universiti Malaysia PAHANG

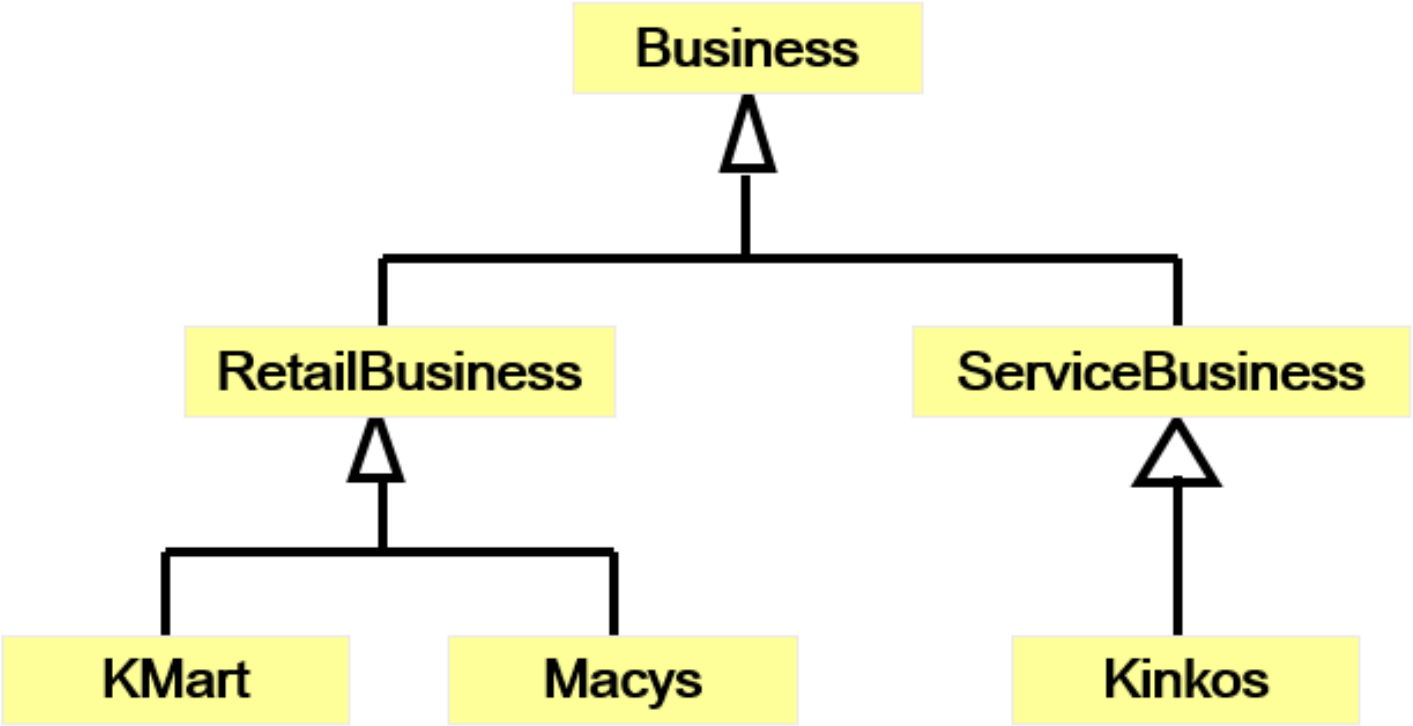
Superclass (Parent)

```
class student
{
int power;
public void setPower (int Power)
…..
}
```

Sub-class (Child)

```
class GradStud extends Student
{
int power; //shadowing variable power
//override the method setPower
Public void setPower (int Power);
int matricNo;     //Student class members
}
```

INHERITANCE : Class Hierarchies



A class hierarchy is form when a child class of one parent become a parents if another child

INHERITANCE : Indirect use of Members

Universiti Malaysia PAHANG

An inherited member can be referenced directly by name in the child class, as if it were declared in the child class

But even if a method or variables is not inherited by a child, it can still **be accessed indirectly** through parent methods.

Member Accessibility : Public, Private and Protected

Will be discussed in the next class….

Give one real-world example that use Inheritance Concept.

- Minimum each parent class has two child
- Draw the Class Diagram for your problem
- Introduce some attributes and method to both parent and child classes

Author Information

Dr. Nor Saradatul Akmar Binti Zulkifli

Senior Lecturer
Faculty of Computer Systems & Software Engineering