QS STARS RATED FOR EXCELLENCE 2012

FPT Education

FPT UNIVERSITY

ĐẠI HỌC FPT CẦN THƠ

Interface and Inheritance

Session 5

Objectives

Interfaces

- Describe
- Implement
- Explain the purpose

Abstraction

- Describe
- Implement

Inheritance

- Describe
- Explain the types
- Polymorphism

Interface

- An ***interface*** is a reference type, similar to a class, that can contain *only* constants, method signatures, default methods, static methods, and nested types. Method bodies exist only for default methods and static methods. Interfaces cannot be instantiated—they can only be *implemented* by classes or *extended* by other interfaces. Extension is discussed later in this lesson.

Interface

- Declare:
  - All the methods in an interface are declared with the empty body
  - All the fields are public, static and final by default.
  - A class that implements an interface must implement all the methods declared in the interface.

```
interface <interface_name>{

    // declare constant fields
    // declare methods that abstract
    // by default.
}
```

Implementation interfaces

```
interface printable{
void print();
}

class Student implements printable{
    private String name;
    private char gender;
    private int year_of_birth;
    private float GPA;
    public void initialize(){...6 lines }
    public void enroll(){...9 lines };
    public void exam(float GPA){...3 lines }
    public void print(){
        System.out.print
        System.out.print
        System.out.print
        System.out.print
        System.out.print
    };
}
```

```
public class Example {
    public static void main(String[] args) {
        Student objStudent = new Student();
        objStudent.enroll();
        objStudent.exam(5);
        objStudent.print();
    }
}
```

Implementation interfaces

```
6   package example;
7
8   import java.util.Scanner;
9   interface printable{
10  void print();
11  }
12  Student is not abstract and does not override abstract method print() in printable
13  ----
14  (Alt-Enter shows hints)
    class Student implements printable{
16      private String name;
17      private char gender;
18      private int year_of_birth;
19      private float GPA;
20      public void initialize(){...6 lines }
26      public void enroll(){...9 lines };
35      public void exam(float GPA){...3 lines }
38  }
```

The purpose of interfaces

- It is used to achieve abstraction.
- By interface, we can support the functionality of multiple inheritance.
- It can be used to achieve loose coupling.

Student Activities

Employee
- name: string - year_of_birth: int - salary: int
- **entry**() - …. - **print**()

Student
- name: string - gender: char - year_of_birth: int - GPA: float
- **entry**() - …. - **print**()

Output
- print()

Input
- entry()

Output, Input are Interface to set names of printable and entry data functions

Main:

declare 1 employee, recruit and print his info

declare 1 student, enroll and print his info

Java 8 Default Method in Interface

- Since Java 8, we can have method body in interface. But we need to make it default method

```
interface Printable{
    void print();
    default void Msg(){
    System.out.println("---- Student Info ---- ");
    }
}
```

```
public class Example {
    public static void main(String[] args) {
        Student objStudent = new Student();
        objStudent.enroll();
        objStudent.exam(5);
        objStudent.Msg();
        objStudent.print();
    }
}
```

Objectives

Interfaces

- Describe
- Implement
- Explain the purpose

Abstraction

- Describe
- Implement

Inheritance

- Describe
- Explain the types
- Polymorphism

Abstraction

- Abstraction shows only essential things to the user and hides the internal details
- Abstraction lets you focus on what the object does instead of how it does I

- There are two ways to achieve abstraction in java
  - Abstract class
  - Interface

Abstract class

- It can have abstract and non-abstract methods.
- It needs to be extended and its method implemented.
- It cannot be instantiated.

Implementation

```
abstract class Person{
  abstract void print();
  abstract void entry();
}

class Student extends Person{
    private String name;
    private char gender;
    private int year_of_birth;
    private float GPA;
    public void initialize(){...6 lines }
    public void entry(){...9 lines };
    public void exam(float GPA){...3 lines }
    public void print(){...6 lines };
}

public class Example {
    public static void main(String[] args) {
        Student objStudent = new Student();
        objStudent.entry();
        objStudent.exam(5);
        objStudent.print();
    }
}
```

Objectives

Interfaces

- Describe
- Implement
- Explain the purpose

Abstraction

- Describe
- Implement

Inheritance

- Describe
- Explain the types
- Polymorphism

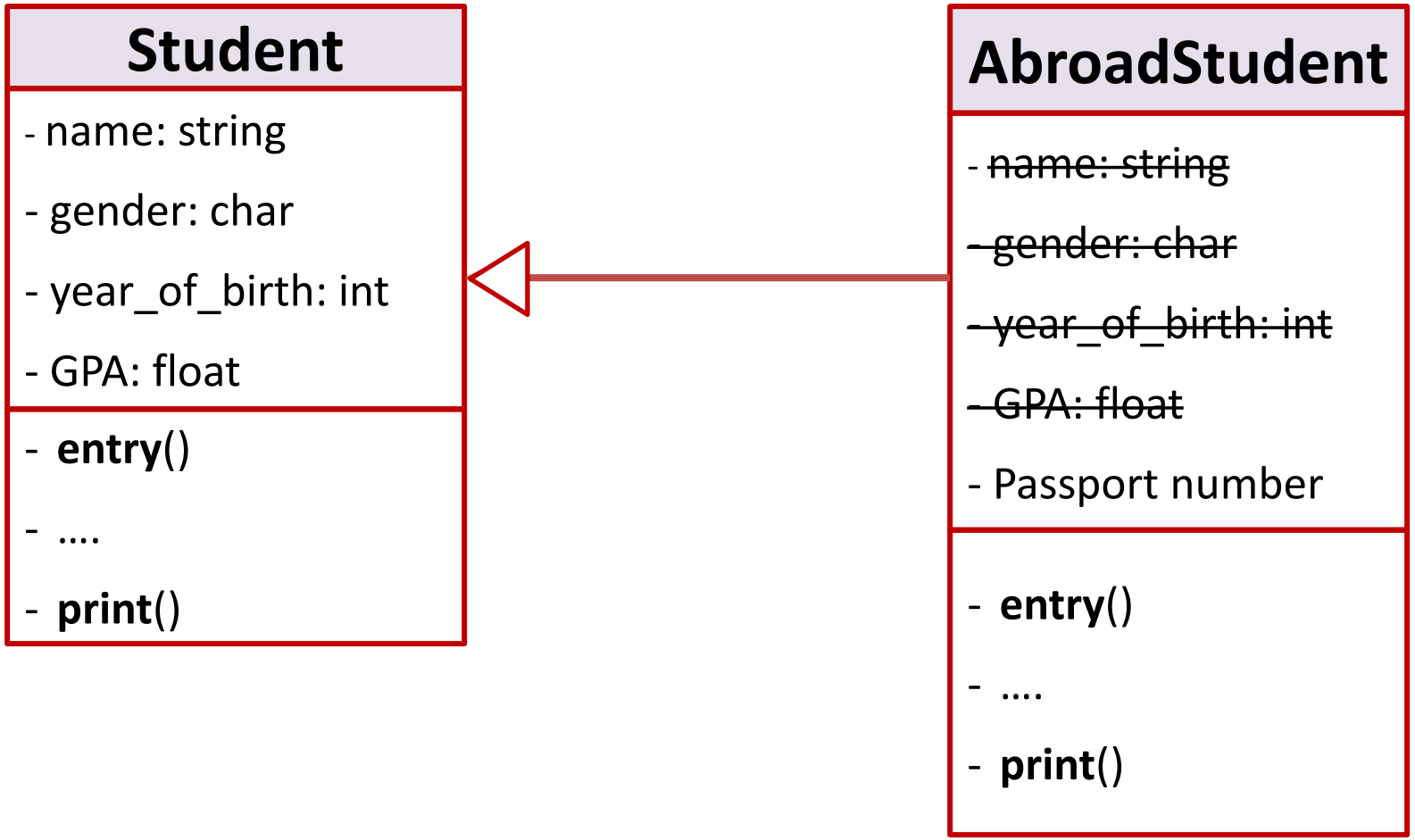
Inheritance

- Classes can be *derived* from other classes, thereby *inheriting* fields and methods from those classes.

- ***Superclass:*** which the subclass is derived
- ***Subclass:*** is derived from another class, can **reuse** the fields and methods of superclass without having to re-write or debug the code again (except private)

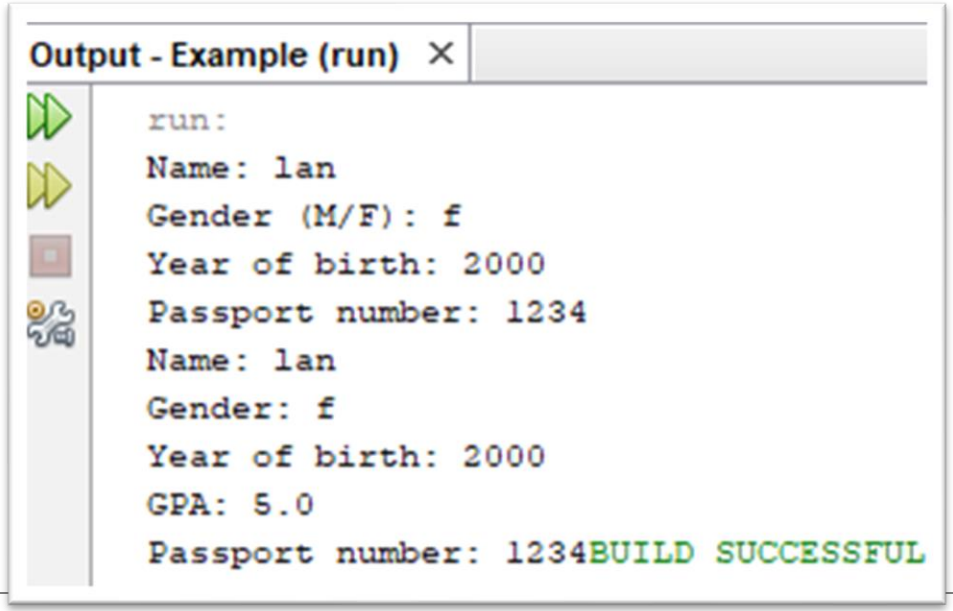
```
class Subclass-name extends Superclass-name
{
  //methods and fields
}
```

Inheritance



```
class Student {
    private String name;
    private char gender;
    private int year_of_birth;
    private float GPA;
    public void initialize() {...6 lines }
    public void entry() {...9 lines };
    public void exam(float GPA) {...3 lines }
    public void print() {...6 lines };
}
class AbroadStudent extends Student{
    String passport;
    public void entry(){
        Scanner scanner = new Scanner(System.in);
        super.entry();
        System.out.print("Passport number: ");
        passport = scanner.next();
    };
    public void print(){
         super.print();
         System.out.print("Passport number: "+passport);
    };
}
```

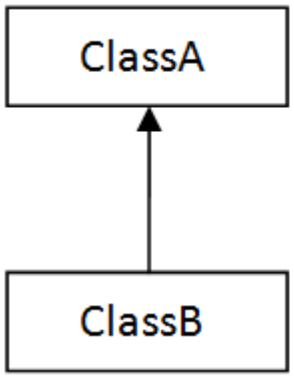
```
public class Example {
    public static void main(String[] args) {
        AbroadStudent objStudent = new AbroadStudent();
        objStudent.entry();
        objStudent.exam(5);
        objStudent.print();
    }
}
```



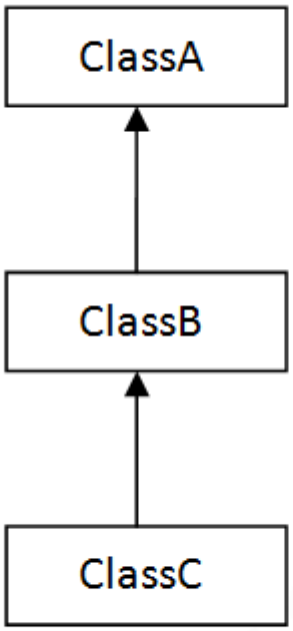
What You Can Do in a Subclass

- The inherited fields / methods can be used directly, just like any other fields / methods (except private fields)
- You can write a new variable / static method in the subclass that has the same signature as the one in the superclass, thus **hiding** it.
- You can write a new instance method in the subclass that has the same signature as the one in the superclass, thus **overriding** it.
- You can write a subclass constructor that invokes the constructor of the superclass, either implicitly or by using the keyword `super`.

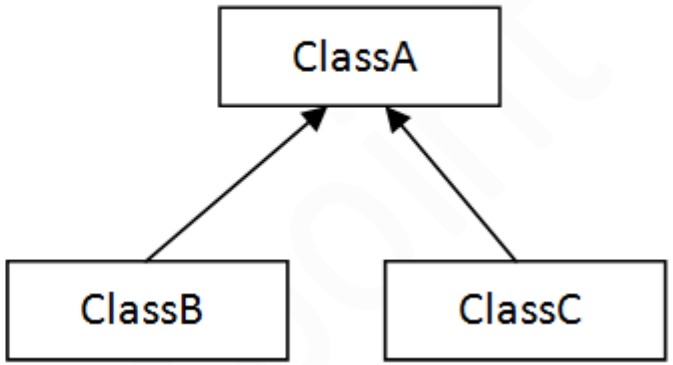
Types of inheritance in java



1) Single



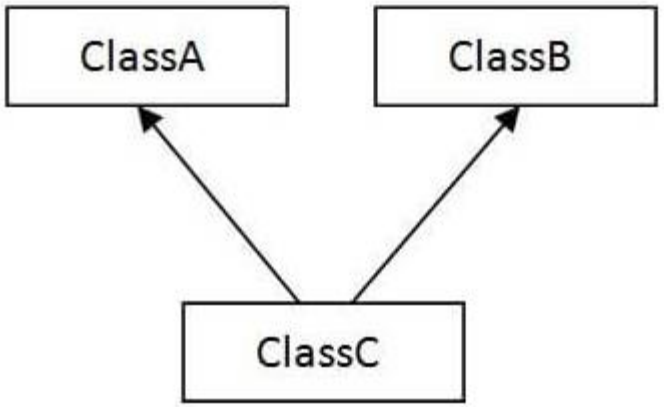
2) Multilevel



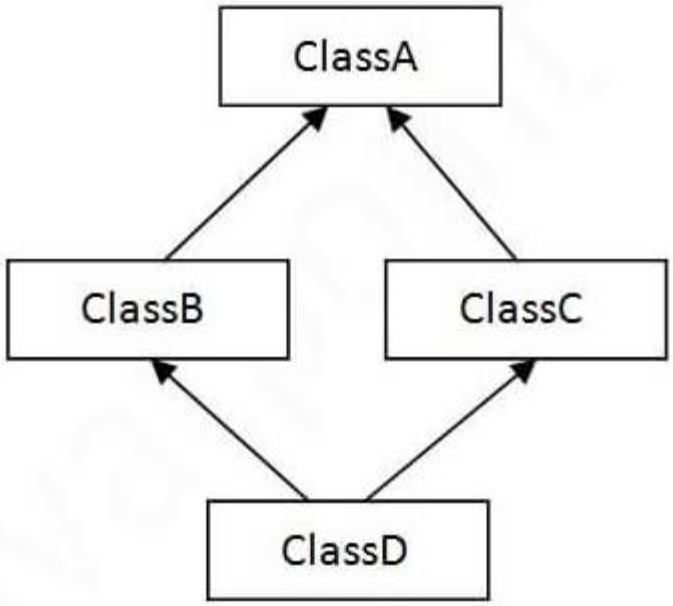
3) Hierarchical

Types of inheritance in java

- Multiple inheritance is **not supported** in Java through class.



4) Multiple



5) Hybrid

Polymorphism

- Polymorphism means many forms.

```
public class Example {
    public static void main(String[] args) {
        AbroadStudent objStudent = new AbroadStudent();
        objStudent.print();
        Student objStudent1 = new Student();
        objStudent1.print();
    }
}
```

Student Activities

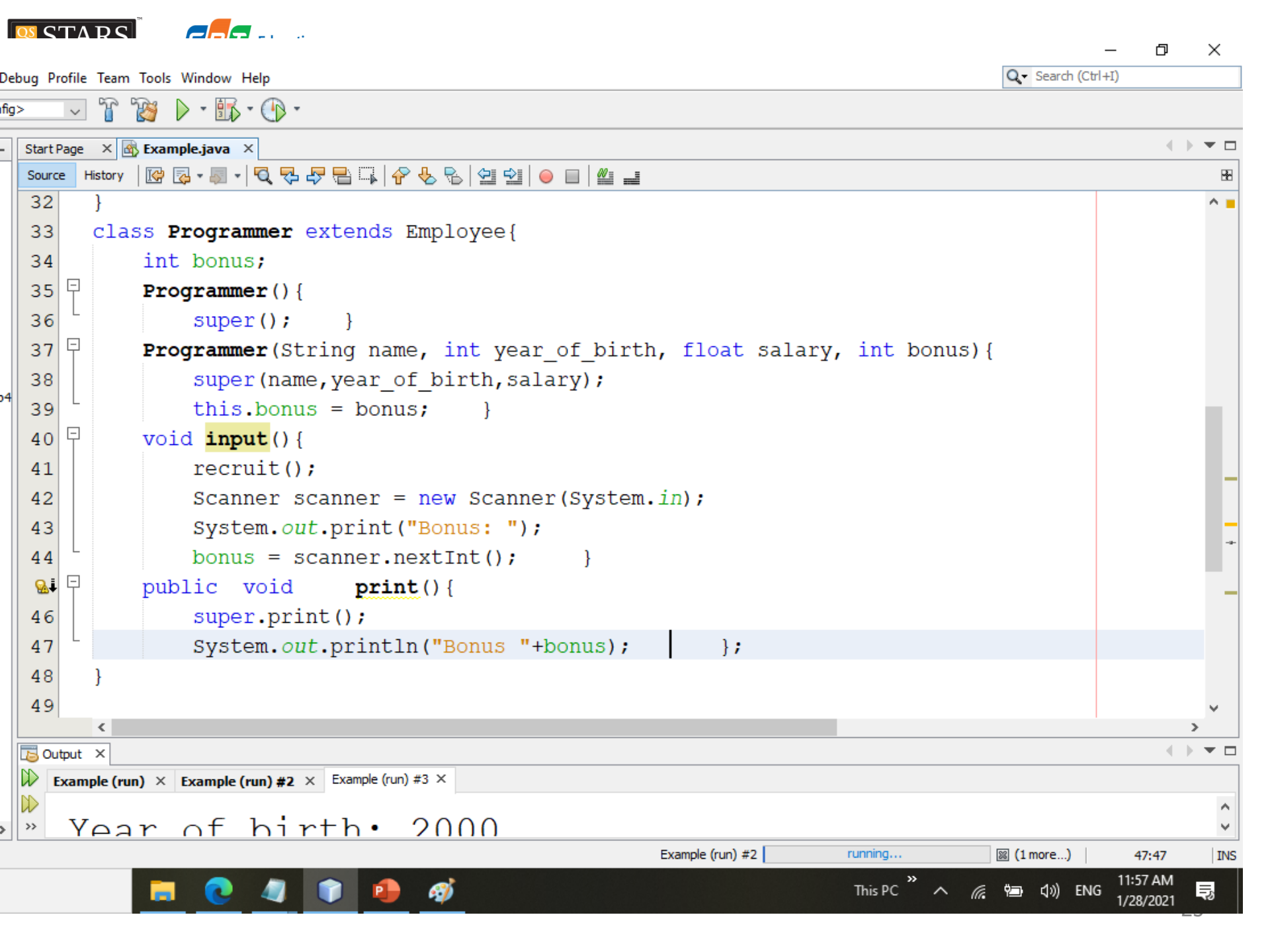
Employee
- name: string - year_of_birth: int - salary: int
- **entry**() - …. - **print**()

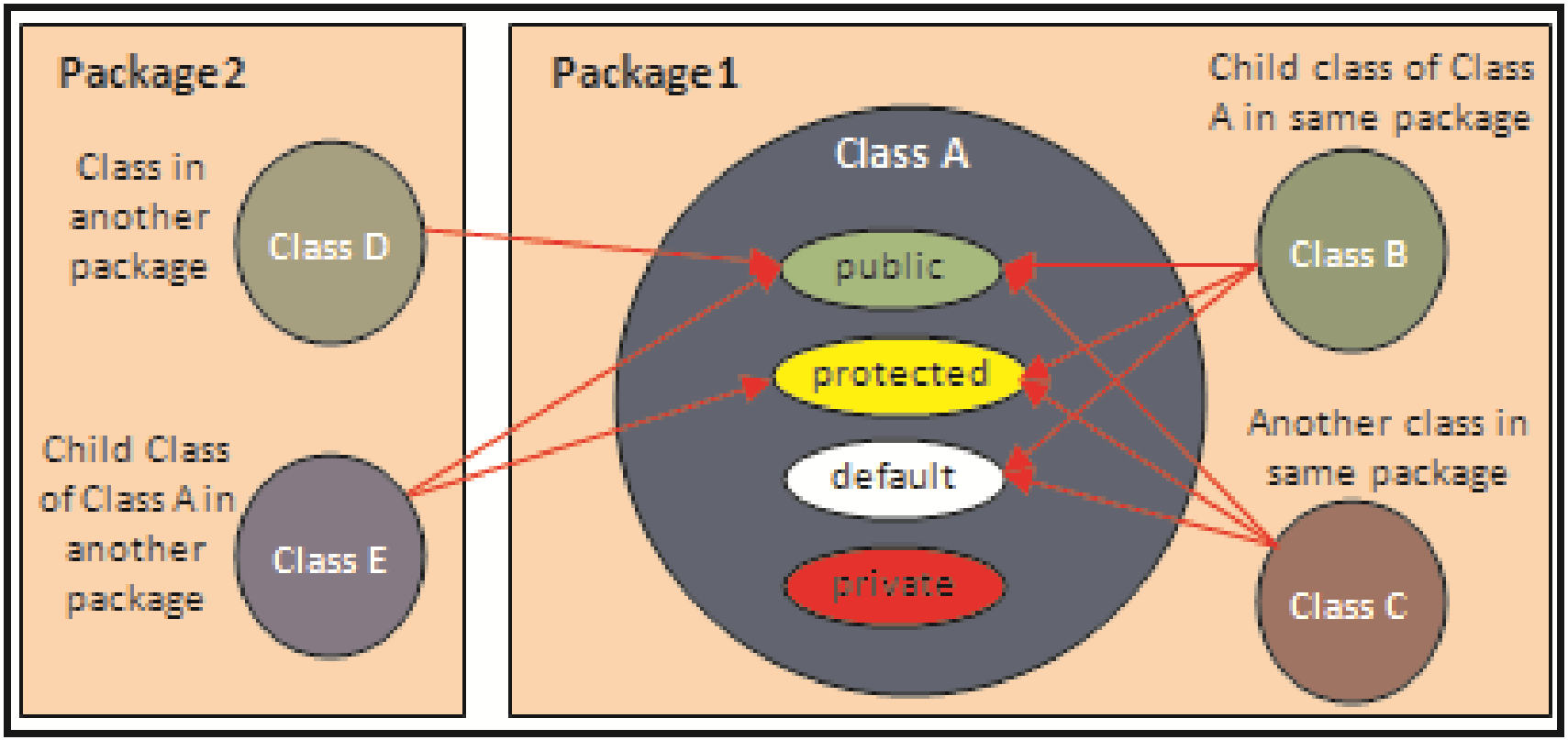
Programmer
- name: string - year_of_birth: int - salary: int - bonus:int
- **entry**() - …. - **print**()

Reuse Employee class, define Programmer class

Main:

Declare 1 programmer, entry and print his info


Debug Profile Team Tools Window Help
Search (Ctrl+I)
Start Page
Example.java
Source
History
32 }
33 class Programmer extends Employee{
34 int bonus;
35 Programmer(){
36 super(); }
37 Programmer(String name, int year_of_birth, float salary, int bonus){
38 super(name,year_of_birth,salary);
39 this.bonus = bonus; }
40 void input(){
41 recruit();
42 Scanner scanner = new Scanner(System.in);
43 System.out.print("Bonus: ");
44 bonus = scanner.nextInt(); }
public void print(){
46 super.print();
47 System.out.println("Bonus "+bonus); };
48 }
49
Output
Example (run)
Example (run) #2
Example (run) #3
Year of birth: 2000
Example (run) #2
running...
(1 more...)
47:47
INS
This PC
ENG
11:57 AM
1/28/2021


Package2
Class in another package
Class D
Child Class of Class A in another package
Class E
Package1
Class A
public
protected
default
private
Child class of Class A in same package
Class B
Another class in same package
Class C

Principles of OOP

- **Object** – Represents an entity which possesses certain features and behaviors.
- **Class** – Is a template that is used to create objects of that class.
- **Abstraction** – Is a design technique that focuses only on the essential features of an entity for a specific problem domain.
- **Encapsulation** – Is a mechanism that combines data and implementation details into a single unit called class.
- **Inheritance** – Enables the developer to extend and reuse the features of existing classes and create new classes. The new classes are referred to as derived classes.
- **Polymorphism** – Is the ability of an object to respond to same message in different ways.

ĐẠI HỌC FPT CẦN THƠ

ĐẠI HỌC FPT CẦN THƠ