CS202: IT Workshop

Java

Inheritance

Ref:

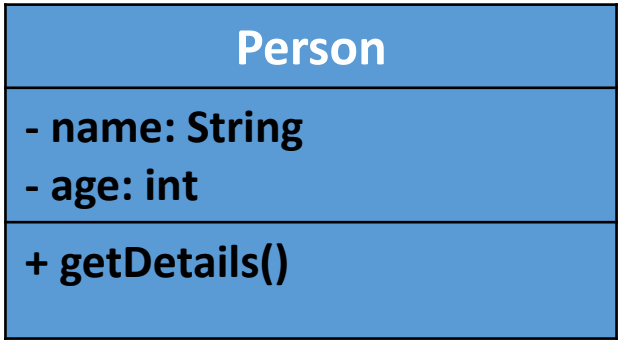
1. Harvey Deitel, Paul Deitel, **Java: How to Program**, 9/e, Prentice Hall India.
2. Herb Schildt, **Java: The Complete Reference**, 8/e Tata Mcgraw Hill Education.

Programming related help

- It is **good** to write programs without taking much help
- But, if you face problem, do not hesitate to approach
  - Ask your doubt in private/public chat in Codetantra
  - If needed, we will make you a **presenter**
  - You can share your screen and get your problems solved

What have we learned in last lecture?

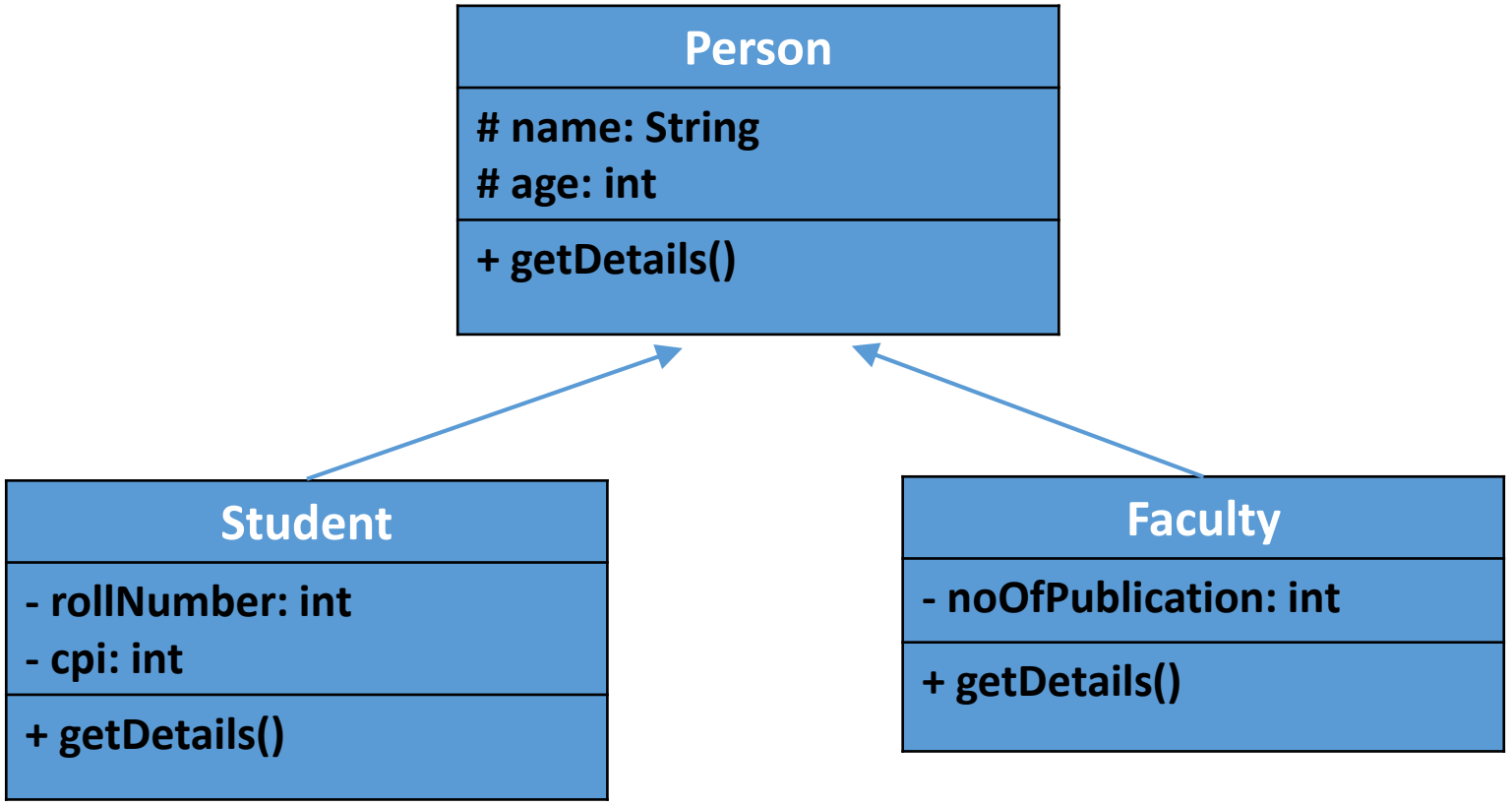
- Object oriented features: Class, Object, Encapsulation
- Access specifiers of members of class
- Constructors and its usages
- Static variables, methods and its usages
- Diagrammatic representation of class: UML class diagram



Inheritance

- **Motivation**: SemiconductorDevice → → Diode
- Class inherits properties from its ancestor
- A new class (called subclass) is created by absorbing existing class's (called super class) members and embellishing them with new or modified capabilities
  - **Reuse** the existing code and feature of super class
  - Embellishment is done by adding new attributes and/or adding/modifying methods
- Inheritance can also be referred to as specialization (super → sub) or generalization (sub → super)
- In C++, super class is known as base class and subclass is known as derived class

Inheritance example




```
class Student extends Person {
. . . . .
}
```

Student will have name, age and rollNumber, cpi

Faculty will have name, age and noOfPublication

```
class Faculty extends Person {
. . . . .
}
```

Inheritance

- All the classes (Person, Student and Faculty) have method **getDetails()**
- Will the code for method **getDetails()** be same for all the classes?

Inheritance: Method overriding

- **getDetails()** in Person only displays name and age
  - **getDetails()** in Student needs to display roll number also
  - **getDetails()** in Faculty needs to display no of publication

- Subclass can redefine a method to do specific work
  - The **signature** of the method has to be **same**

```
void getDetails() { … }
   . . .

void getDetails( int i) { … }
```

Signature is NOT same

X

- This is called **method overriding**

Inheritance: Method overriding example

```
public class Student extends Person {
    . . .
    @Override
    void getDetails() {

            System.out.println("Name: + name + " Age: " + age + " Roll: " + rollNumber);
            }
}
```

Subclass may completely write new codes

```
public class Student extends Person {
    . . .
    @Override
    void getDetails() {

            super.getDetails();
            System.out.println(" Roll: " + rollNumber);
            }
}
```

Subclass may also reuse the implementation of the superclass method

@Override annotation may be used to avoid unintentional errors; Compiler matches the method signature with that of superclass

Inheritance: Method overriding example

- A subclass can call the implementation of the method of its superclass using a **dot operator**
  e.g. super.**getDetails()** [Please refer last slide]

```
Person p = new Person();
p.getDetails();

Student s = new Student();
s.getDetails();

Faculty f = new Faculty();
f.getDetails();
```

We have 3 implementations of method getDetails()

Based on the calling object (p, s, f), the appropriate method body will be executed.

Code (BoxDemo.java): Screen share

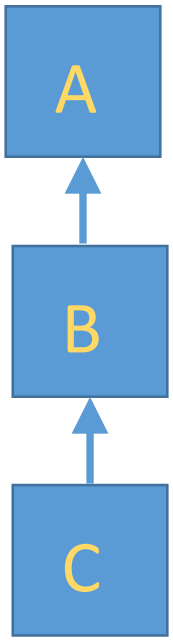
Questions?

Inheritance: Constructors

- Superclass's constructor should be called from the subclass's constructor using **super()**
- This has to be the first line of code in subclass's constructor body
- Default constructors are called implicitly (**super →sub**)

```
class Student extends Person {
   . . .
      Student (String name, int age, int roll) {

            super( name, age );
            . . .
            }
    . . .
}
```



Execution order:

A() → B() → C()

```
C cObj = new C();
```

Code (ConstructorDemo.java): Screen share

Questions?

Polymorphism in Java

- **Polymorphism:**
  - Different forms (or morphs) of same species [Biology]
  - Different forms (definitions) with same method name

- ***getDetails()*** method Person, Student, Faculty
  - Depending on the calling object, different tasks are performed.
  - Type of the object is determined at **runtime** and appropriate body of the method is invoked (**dynamic binding**)

(Associating a method call with its body is called **binding)**

Polymorphism in Java

- Polymorphism was achieved among different classes (Person, Student, Faculty) using **method overriding**

- Polymorphism may also be achieved within the same class using **method overloading**
  - methods with same name but different **signature**

```
double distance ( int x1, int y1) {
            . . .
   }

double distance (Point p) {
            . . .
   }
```

```
obj. distance ( 2, 3 );
```

```
obj. distance ( p );
```

Code (PolymorphDemo.java): Screen share

Questions?