

**Muhammad Shaheer | FA24-BSE-104(B)**

**Object Oriented Programming**

Sir Ameer Gillani



COMSATS UNIVERSITY ISLAMABAD, SAHIWAL

# ***Demonstration of Abstraction, Inheritance and Generics in Java***

23 November 2025

### ASSIGNMENT#03

1. CREATE AN ABSTRACT CLASS "PERSON", WITH DATA MEMBER "NAME". CREATE SET AND GET METHODS, AND AN ABSTRACT BOOLEAN METHOD "ISOUTSTANDING()". DERIVE TWO CLASSES STUDENT AND PROFESSOR. STUDENT CLASS HAS DATA MEMBER CGPA. PROFESSOR CLASS HAS DATA MEMBER NUMBEROFPUBLICATIONS. PROVIDE SETTERS AND GETTERS AND IMPLEMENTATION OF ABSTRACT FUNCTION IN BOTH CLASSES. IN STUDENT CLASS ISOUTSTANDING() WILL RETURN TRUE IF CGPA IS GREATER THAN 3.5. IN THE PROFESSOR CLASS ISOUTSTANDING() WILL RETURN TRUE, IF NUMBEROFPUBLICATIONS> 50. IN THE MAIN CLASS CREATE AN ARRAY OF PERSON CLASS AND CALL ISOUTSTANDING() FUNCTION FOR STUDENT AND PROFESSOR. ISOUTSTANDING() FOR PROFESSOR SHOULD BE CALLED AFTER SETTING THE PUBLICATION COUNT TO 100.

```
abstract class Person{
    private String name;

    Person(){

    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public abstract boolean isOutstanding();
}
class Student extends Person{
    private double CGPA;

    public double getCGPA() {
        return CGPA;
    }

    public void setCGPA(double CGPA) {
        this.CGPA = CGPA;
    }

    @Override
    public boolean isOutstanding() {
        if (CGPA>3.5){
```

```

        return true;
    }
    return false;
}
}

class Professor extends Person{
    private int numberOfPublications;

    public int getNumberOfPublications() {
        return numberOfPublications;
    }

    public void setNumberOfPublications(int
numberOfPublications) {
        this.numberOfPublications = numberOfPublications;
    }

    @Override
    public boolean isOutstanding(){
        if (numberOfPublications> 50){
            return true;
        }
        return false;
    }
}

public class Main {
    public static void main(String[] args) {
        Student st1 = new Student();
        Student st2 = new Student();
        Professor p1 = new Professor();
        Professor p2 = new Professor();

        st1.setName("Shaheer");
        st1.setCGPA(3.5);
        st2.setName("Ali");
        st2.setCGPA(3.7);

        p1.setName("Sir Ameer");
        p1.setNumberOfPublications(89);
        p2.setName("Sir Shehzad");
        p2.setNumberOfPublications(99);

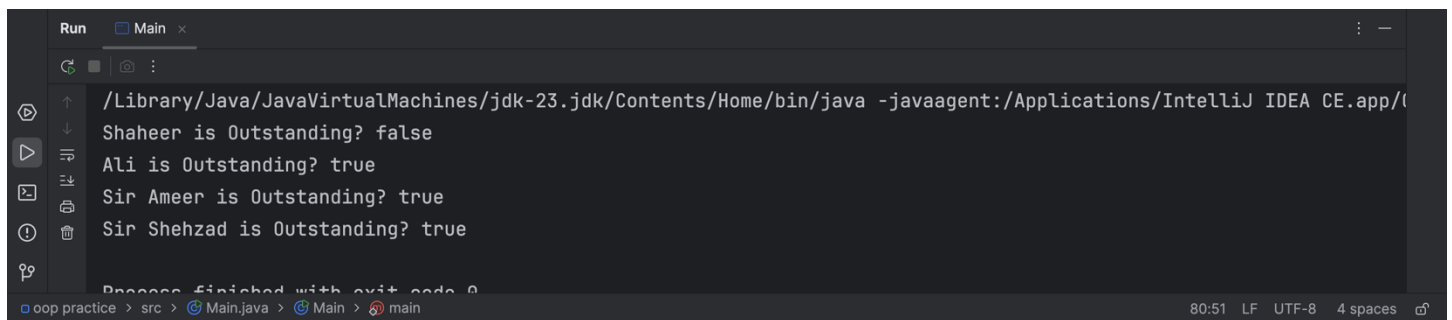
        Person[] person = new Person[4];
    }
}

```

```
    person[0] = st1;
    person[1] = st2;
    person[2] = p1;
    person[3] = p2;

    for (Person p : person) {
        System.out.println(p.getName() + " is Outstanding?"
" + p.isOutstanding());
    }
}
```

OUTPUT:



The screenshot shows the IntelliJ IDEA Run console. The top bar indicates the file being run is 'Main'. The console output shows the following lines:

```
/Library/Java/JavaVirtualMachines/jdk-23.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA CE.app/...
Shaheer is Outstanding? false
Ali is Outstanding? true
Sir Ameer is Outstanding? true
Sir Shehzad is Outstanding? true
Process finished with exit code 0
```

The bottom status bar shows the file path 'oop practice > src > Main.java > Main > main' and the encoding '80:51 LF UTF-8 4 spaces'.

2. CREATE A GENERIC CLASS WITH A TYPE PARAMETER THAT SIMULATES DRAWING AN ITEM AT RANDOM OUT OF A BOX. THIS CLASS COULD BE USED FOR SIMULATING A RANDOM DRAWING. FOR EXAMPLE, THE BOX MIGHT CONTAIN STRINGS REPRESENTING NAMES WRITTEN ON A SLIP OF PAPER, OR THE BOX MIGHT CONTAIN INTEGERS REPRESENTING A RANDOM DRAWING FOR A LOTTERY BASED ON NUMERIC LOTTERY PICKS. CREATE AN ADD METHOD THAT ALLOWS THE USER OF THE CLASS TO ADD AN OBJECT OF THE SPECIFIED TYPE ALONG WITH AN ISEMPTY METHOD THAT DETERMINES WHETHER OR NOT THE BOX IS EMPTY. FINALLY, YOUR CLASS SHOULD HAVE ADRAWITEM METHOD THAT RANDOMLY SELECTS AN OBJECT FROM THE BOX AND RETURNS IT. IF THE USER ATTEMPTS TO DRAWN AN ITEM OUT OF AN EMPTY BOX, RETURN NULL. WRITE A MAIN METHOD THAT TESTS YOUR CLASS.

```
import java.util.*;

class Box<T> {
    private ArrayList<T> list = new ArrayList<>();

    public void add(T value) {
        list.add(value);
    }

    public boolean empty() {
        return list.size() == 0;
    }

    public T pickRandom() {
        if (empty()) {
            return null;
        }

        Random r = new Random();
        int randomIndex = r.nextInt(list.size());
        T selected = list.get(randomIndex);

        list.remove(randomIndex);
        return selected;
    }
}

public class Main {
    public static void main(String[] args) {

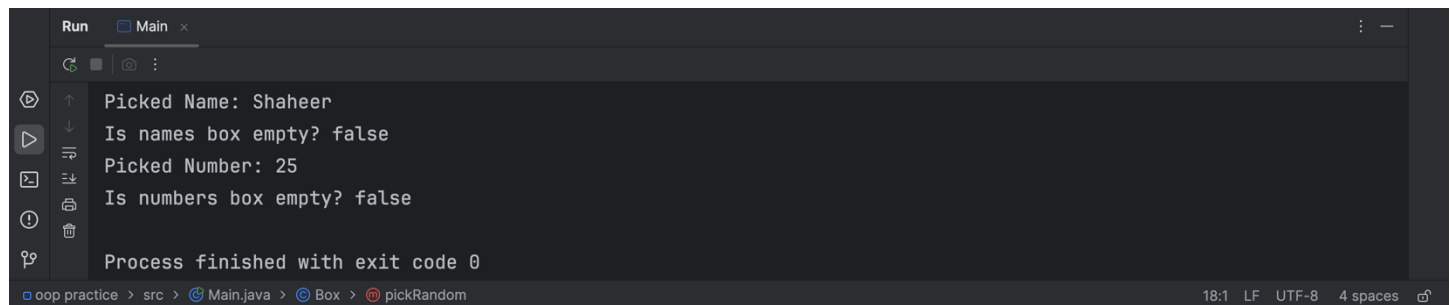
        Box<String> names = new Box<>();
        names.add("Shaheer");
        names.add("Ali");
        names.add("Hamza");
    }
}
```

```
        System.out.println("Picked Name: " +
names.pickRandom());
        System.out.println("Is names box empty? " +
names.empty());

        Box<Integer> numbers = new Box<>();
        numbers.add(5);
        numbers.add(15);
        numbers.add(25);

        System.out.println("Picked Number: " +
numbers.pickRandom());
        System.out.println("Is numbers box empty? " +
numbers.empty());
    }
}
```

#### OUTPUT:



```
Run Main x
Picked Name: Shaheer
Is names box empty? false
Picked Number: 25
Is numbers box empty? false
Process finished with exit code 0
oop practice > src > Main.java > Box > pickRandom 18:1 LF UTF-8 4 spaces
```