Object Oriented Programming with Java
Lab Practice:5

1. Consider the following code:

20

```
public class A {
  public void One(int i) {
  }
  public void Two(int i) {
  }
  public static void Three(int i) {
  }
  public static void Four(int i) {
  }
}
public class B extends A {
  public static void One(int i) {
  }
  public void Two(int i) {
  }
  public void Three(int i) {
  }
  public static void Four(int i) {
  }
}
```

override

hide

Answer the followings:

static method override X ?

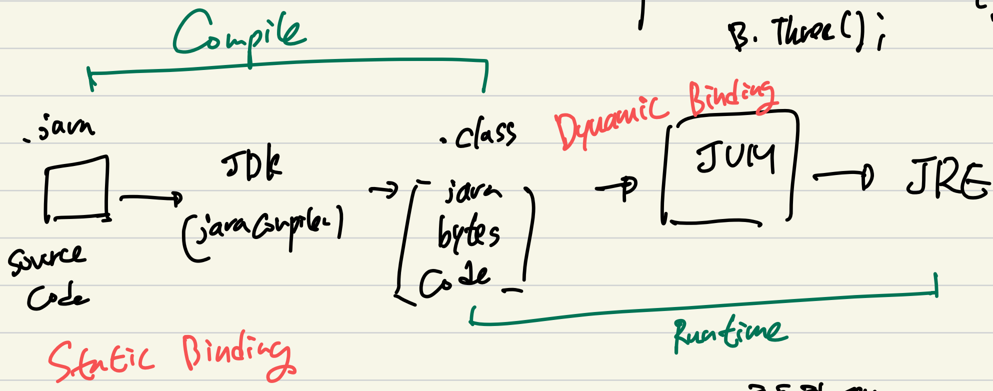
why not?

A a = new A();

a.Three();

B ——→ Three() (인스턴스 메서드)

B.Three();



JVM이 메서드를 호출할 때,
instance method의 경우 해당 메서드를 구현하고 있는 실제 객체를 찾아 호출 (다형성)

But, static method는 컴파일 시점에 선언된 타입의 메서드 호출 (다형성 적용 X)

사실 중요한 점은

Compile 오류가 → 컴파일 시점 오류
발생한 사실이다

됐다고 치면 다형성 따라.

A static void a( )

(X) void a( )

B void a( )

a. Which method overrides a method in the superclass?
Two and Four in B

Two

b. Which method hides a method in the superclass?
Three, Four hid. Using static

Four.

c. What do the other methods do?
Errors.

Compile

2. Create a class name 'Person'. The class contains two fields; String calle firstName and lastName and the following methods.

a. default and alternate constructors in the class.

b. two accessors (getter) to return the first and the last name.

c. A method called setName to set the fields to the parameters passed.

d. A method called print (should print first and last)

e. A method name toString()

f. a method name equals (pass an object of the Object class)

g. two methods name copy and getCopy to make a copy of the Person object into another Person object.

Person

Employee

Note: the Person class is the super class for a class called employee. And this employee class should contain three fields (payRate, workHours, and deptName)
Consider the following structure of code.

```
//Class Employee: subclass of Person
public class Employee extends Person {
   private double payRate;
   private double workHours;
```

```
    private String deptName;

    public final int HOURS = 35;
    public final double OVERTIME = 1.2;

     //default constructor
     public Employee() {
  ...
     }
    //add an alternate constructor with
parameters

     public String toString() {
        //should return a String like this:
         //The wages for xxxx from the xxxx
department are: $xxxxx.xx"
    ...
     }
     public void print() {
       //Should print output like this (same
line):
        //The employee xxxx from the xxxx
department worked xx hours
        //with a pay rate of $xxx.xx. The
wages for this employee are $xxxxx.xx
       ...
     }
     public double calculatePay() {
       //Method to calculate and return the
```

```
wages
            //handle both regular and overtime
pay
              ...
            }
            public void setAll(String first, String
last, double rate, double hours, String dep){
              ...
            }
            public double getPayRate() {
              ...
          }
                public double getHoursWorked()
{
              ...
              }
            public String getDepartment() {
                ...
              }
            public boolean equals(Object o) {
              ...
              }
            public Employee getCopy() {
              ...
            }
            public void copy(Employee e) {
              ...
            }
             }}
```

```
public class Person {
    String firstname;
    String lastname;


    Person() {
    }

    Person(String firstname, String lastname) {
        this.firstname = firstname;
        this.lastname = lastname;
    }

    public String getFirstname() {
        return firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setName(String firstname, String lastname) {
        this.firstname = firstname;
        this.lastname = lastname;
    }

    public void print() {
        //Should print output like this (same line):
        //The employee xxxx from the xxxx department worked
xx hours
        //with a pay rate of $xxx.xx. The wages for this
employee are $xxxxx.xx
        System.out.println("The employee name :" +
this.toString());

    }


    public String toString() {

        return this.firstname + this.lastname;
    }


    public Person getCopy() {

        return new Person(this.getFirstname(),
```

```
this.getLastname());
    }

    public void copy(Person p) {
        this.setName(p.firstname, p.lastname);
    }


    public boolean equals(Object o) {
        if (this == o) {
            return true;
        }

    return true;
    }
}


//Class Employee: subclass of Person
public class Employee extends Person {
    private double payRate;
    private double workHours;
    private String deptName;

    public final int HOURS = 35;
    public final double OVERTIME = 1.2;

    //default constructor
    public Employee() {

    }

    //add an alternate constructor with parameters
    Employee(String firstname, String lastname, double
payRate, double workHours, String deptName) {
        super(firstname, lastname);
        this.payRate = payRate;
        this.workHours = workHours;
        this.deptName = deptName;

    }

    public String toString() {
        //should return a String like this:
        //The wages for xxxx from the xxxx department are:
$xxxxx.xx"

        return "The wages for" + super.toString() + "from
the " + this.deptName + "department are : $" +
this.calculatePay();
```

```
    }

    public void print() {
        //Should print output like this (same line):
        //The employee xxxx from the xxxx department worked
xx hours
        //with a pay rate of $xxx.xx. The wages for this
employee are $xxxxx.xx
        System.out.println("The employee " +
super.toString() + "from the " + this.deptName +
"department worked " +
                this.workHours + "hours with a pay rate of
$" + this.payRate + "The wages for this employee are $" +
this.calculatePay());
    }

    public double calculatePay() {
        //Method to calculate and return the wages
        //handle both regular and overtime pay
        //35 보다 작으면 w = x * h
        // else w = x * 35 + x (h-35)*1.2
        if (this.workHours <= HOURS) {
            return this.workHours * this.payRate;
        } else {
            return (this.payRate * HOURS) + (this.payRate *
(this.workHours - HOURS) * OVERTIME);
        }

    }

    public void setAll(String first, String last, double
rate, double hours, String dep) {
        super.setName(first, last);
        this.payRate = rate;
        this.workHours = hours;
        this.deptName = dep;
    }

    public double getPayRate() {
        return this.payRate;
    }

    public double getHoursWorked() {
        return this.workHours;
    }

    public String getDepartment() {
        return this.deptName;
    }

    public boolean equals(Object o) {
```

```
        if (this == o)
            return true;
        if (!super.equals(o))
            return false;
        if (getClass() != o.getClass())
            return false;
        Employee other = (Employee) o;
        if (HOURS != other.HOURS)
            return false;
        if (deptName == null) {
            if (other.deptName != null)
                return false;
        }
        return true;


    }

    public Employee getCopy() {
        return new Employee(this.getFirstname(),
this.getLastname(), this.getPayRate(),
this.getHoursWorked(), this.getDepartment());

    }

    public void copy(Employee e) {
        this.setAll(e.firstname, e.lastname, e.payRate,
e.workHours, e.deptName);

    }

    public void main(String[] args) {


    }
}
```