데이터융합SW과
김규석 교수

JAVA

기본 프로그래밍 05

Objective

Class

- Object
- Constructor
- this
- Function vs Method
- Inner Class

Interface

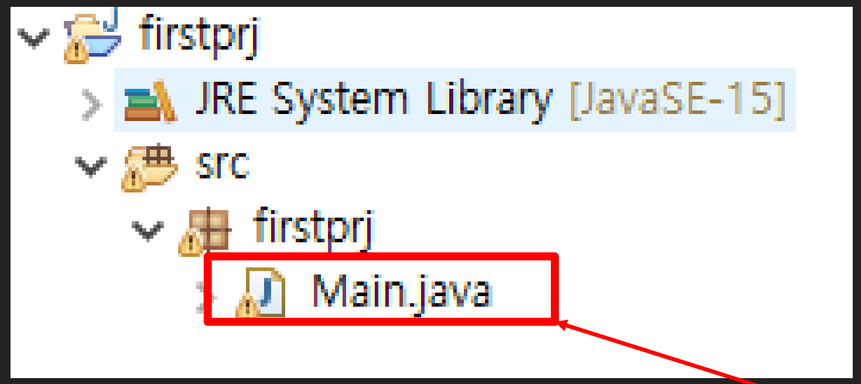
- Inheritance
- super
- Overriding
- Polymorphism

Class(Cont'd)

Object-Oriented Programming

- Java is combined with Classes and Objects

- Ex) A phone is an object, the functions such as Wi-Fi on it are methods



Class

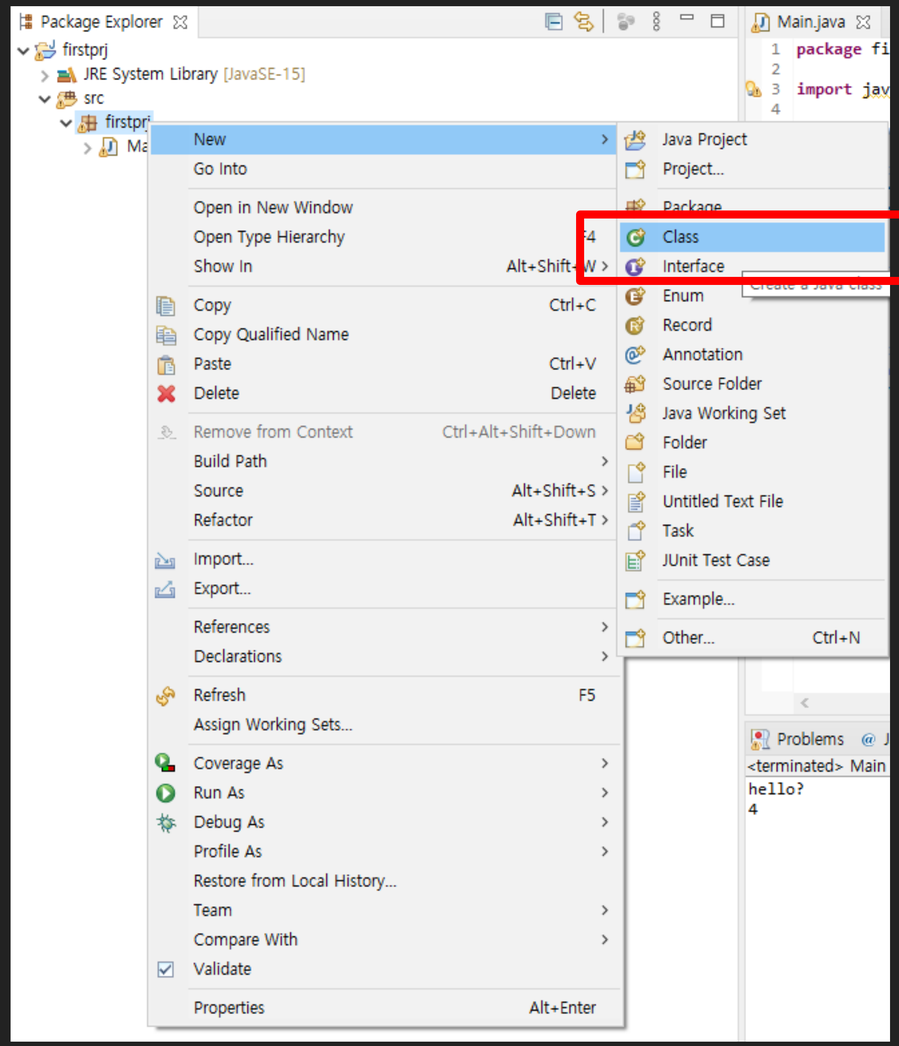
Q1* : What's the difference between PP and OOP?

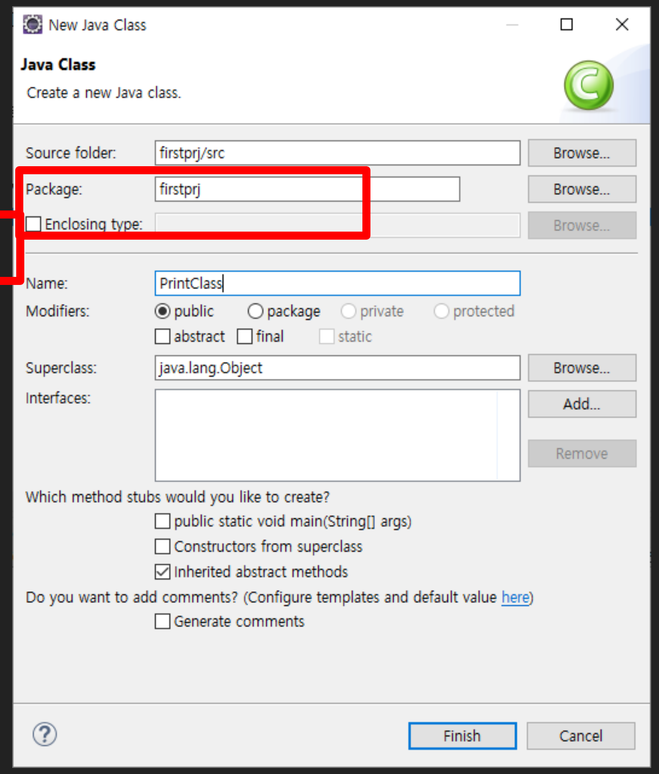
Q2* : What's the difference between "Class" and "Object"?

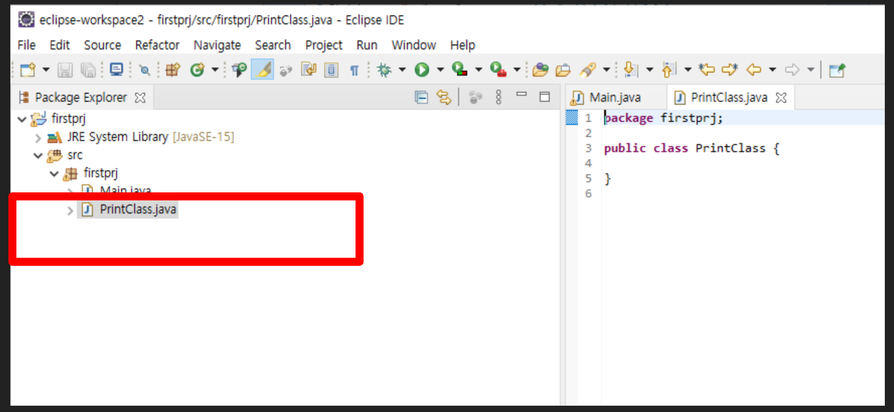
Class(Cont'd)

Create a New Class

- Click on Package Name > Right-click > New > Class







Class(Cont'd)

Create Methods

- In the body of "PrintClass", add some methods

```
3  public class PrintClass {
4      public void printInteger(int n) {
5          System.out.println(n);
6      }
7
8      public void printAdd(int a, int b) {
9          int sum;
10         sum = a + b;
11         System.out.println(sum);
12     }
13 }
```

Q3* : How do we call the functions?

Class(Cont'd)

Create an Object and Call a Method of it

- ClassName ObjectName = new ClassName();

```
5  public class Main {
6
7      public static void main(String[] args) {
8          PrintClass printClass = new PrintClass();
9          printClass.printAdd(1, 2);
10     }
11 }
12
```

Instance of a Class

- A concrete occurrence of any object

Q4* : Explain "Class", "Object" and "Instance".

Q5* : Explain "Method" and "Function".

Class(Cont'd)

Constuctor

- A block of code that initializes the newly created object

```java
3  public class PrintClass {
4      int a, b, n;
5
6      public PrintClass(int a, int b, int n) {
7          // TODO Auto-generated constructor stub
8          this.a = a;
9          this.b = b;
10         this.n = n;
11     }
12
13     public void printInteger() {
14         System.out.println(this.n);
15     }
16
17     public void printAdd() {
18         int sum;
19         sum = this.a + this.b;
20         System.out.println(sum);
21     }
22 }
```

```java
5  public class Main {
6
7      public static void main(String[] args) {
8          PrintClass printClass = new PrintClass(1, 2, 5);
9          printClass.printInteger();
10         printClass.printAdd();
11     }
12 }
```

Constructor

Problems @ Javadoc Declaration Console

<terminated> Main [Java Application] C:₩Users₩CTC₩.p2₩po

```
5
3
```

P1

Constructor Example

- Price
  1. Apple
  2. Strawberry
  3. Grape
  4. Watermelon

- User selects a number

- The each operation should be done in separate functions

- The name of the functions should be the same

- The calculated value should be a return value

Class(Cont'd)

this

- Refers to the current object in a method or constructor

```
1  package test;
2
3  public class Character {
4      String name;
5      int age;
6      int power;
7
8      public Character(String name, int age, int power) {
9          this.name = name;
10         this.age = age;
11         this.power = power;
12     }
13 }
```

P2

Creating Game Characters

- Refer to the example for “this”
- Create a class which has a constructor
- The constructor should have the parameters of “name”, “age”, “offense power” and “defense power”
- Create more than 3 characters with using the character class above
- Print the character introduction

1. A / 200 / 30.5 / 32.1
2. B / 123 / 47.1 / 18.9
3. C / 765 / 21.6 / 42.3

Class(Cont'd)

this()

- Invoke the instructor of the current class

```
1  package test;
2
3  public class Character {
4      String name;
5      int age;
6      int power;
7
8      public Character() {
9          this("hello", 10, 20);
10     }
11
12     public Character(String name, int age, int power) {
13         this.name = name;
14         this.age = age;
15         this.power = power;
16     }
17 }
```

P3

Creating Game Characters II

- Adding some condition code block to P2

- Add two more constructors which have one parameter and two parameters respectively

- Print the character introduction

1. D / 260 / 35.5 / 42.1
2. E / 1213 / 46.1 / 38.9

Method

Access Modifiers

- Public, Private, Protected
- Public : Accessible from everywhere
- Private : Accessible within the same class only
- Protect : Accessible by the classes of the same package

P4 : Explain the meaning of the 12nd line

Class

Inner Class

- Classes Nested in a Class

Main.java

```java
package firstprj;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        PrintClass printClass = new PrintClass(1, 2, 5);
        printClass.printInteger();
        printClass.printAdd();
    }
}

class PrintClass {
    int a, b, n;

    public PrintClass(int a, int b, int n) {
        // TODO Auto-generated constructor stub
        this.a = a;
        this.b = b;
        this.n = n;
    }

    public void printInteger() {
        System.out.println(this.n);
    }

    public void printAdd() {
        int sum;
        sum = this.a + this.b;
        System.out.println(sum);
    }
}
```

Problems | Javadoc | Declaration | Console

<terminated> Main [Java Application] C:₩Users₩CTC₩.p2₩po

```
5
3
```

P5

Inner Class Example

- Compose a program without creating an object

- (Hint, “static”)

P6

Regarding as P3

- Change the created class of “P3” to an inner class

- The other conditions are the same

P7

Descriptive Statistics

- Make a record statistics for students

- Format : Name, Korean Score, English Score, Math Score

- Output : Number of students, average, minimum and maximum scores for each subject

#Menu
1. Input a record
2. Make descriptive statistics
3. Print all the records

1
Kim, 75, 100, 78
2
Number of Students : 5
Korean(Avg, Min, Max) : 75.5, 68.5, 100

Interface(Cont'd)

Inheritance

- Inherits attributes and method from one class to another
- Superclass : the class being inherited from
- Subclass : the class which inherits from another class

```
13  public class Main extends Character {
14      public static int power = 10;
15
16      public static void main(String[] args) {
17          // TODO Auto-generated method stub
18          Main main = new Main();
19          main.printPower();
20      }
21
22      public void printPower() {
23          super.printPower();
24          System.out.println(power);
25      }
```

```
3   public class Character {
4       public static int power = 30;
5
6       public void printPower() {
7           System.out.println(power);
8       }
9   }
```

Problems | Javadoc | Declaration | Console

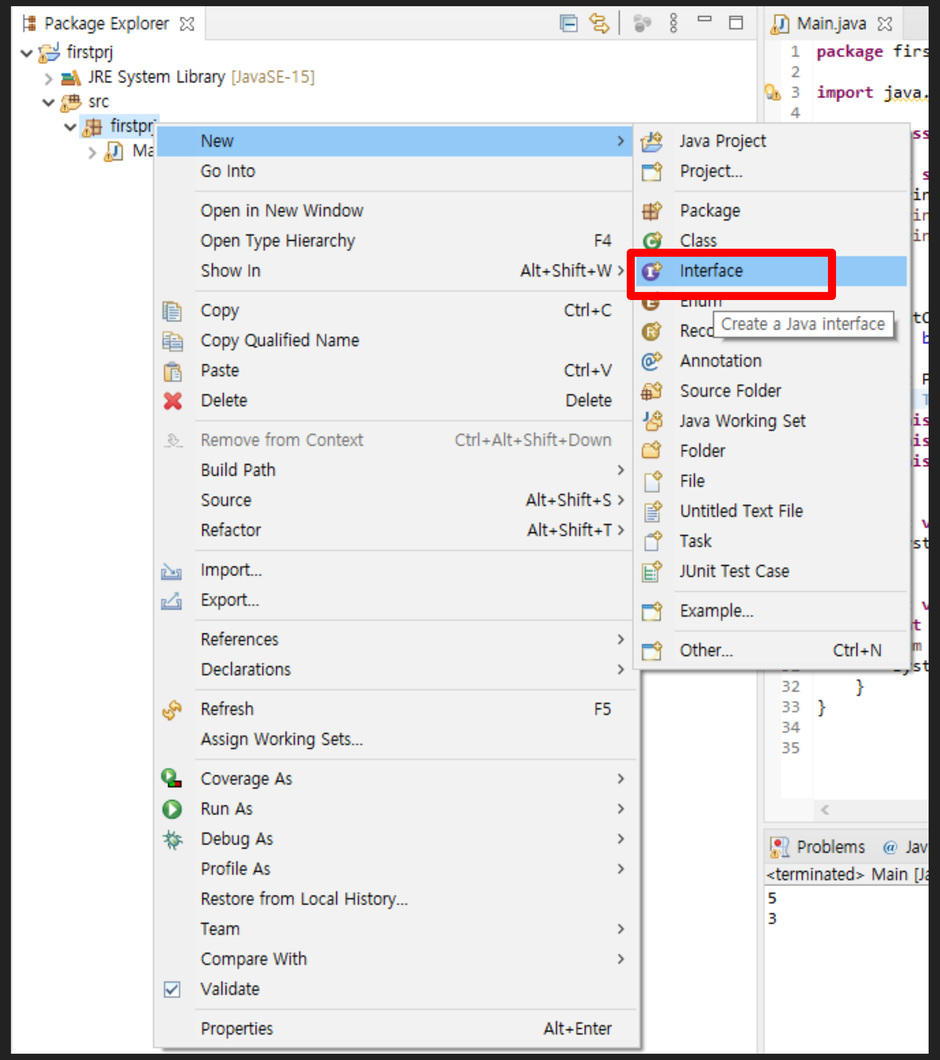
<terminated> Main [Java Application] C:₩Users₩kopo₩.p2₩p

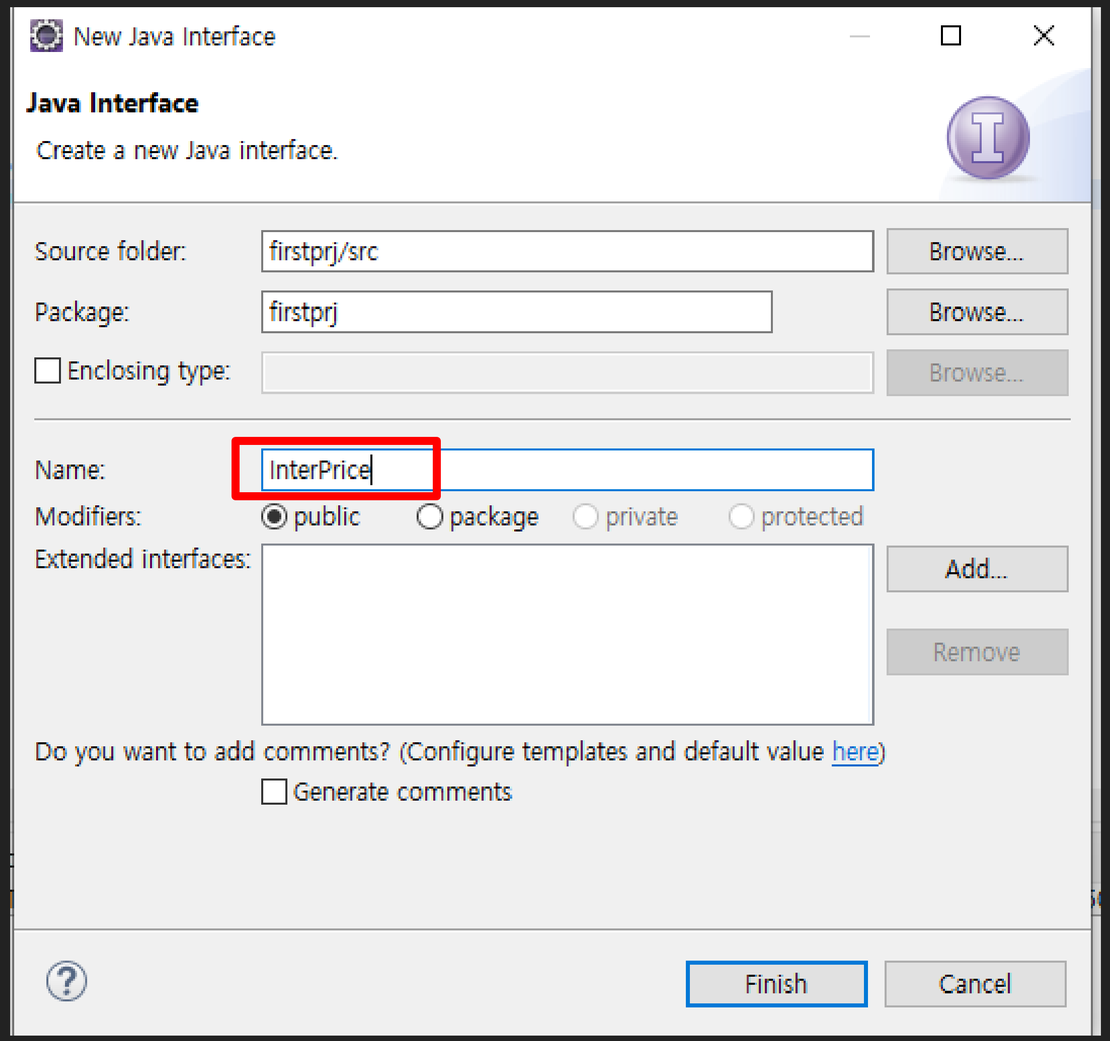
```
30
10
```

Interface(Cont'd)

Interfaces

- Abstract class with empty bodies,





Interface(Cont'd)

Interfaces

- A Group of Related Methods

```java
public class Main implements InterPrice {

    public static void main(String[] args) {
        Main m = new Main();
        m.getPriceA(100);
        m.getPriceB(100);
    }

    @Override
    public void getPriceA(int price) {
        // TODO Auto-generated method stub
        System.out.println(price + 100);
    }

    @Override
    public void getPriceB(int price) {
        // TODO Auto-generated method stub
        System.out.println(price + 1000);
    }
}
```

```java
public interface InterPrice {
    public void getPriceA(int price);
    public void getPriceB(int price);
}
```

Problems | Javadoc | Declaration | Console

<terminated> Main [Java Application] C:₩Users₩CTC₩.p2₩p

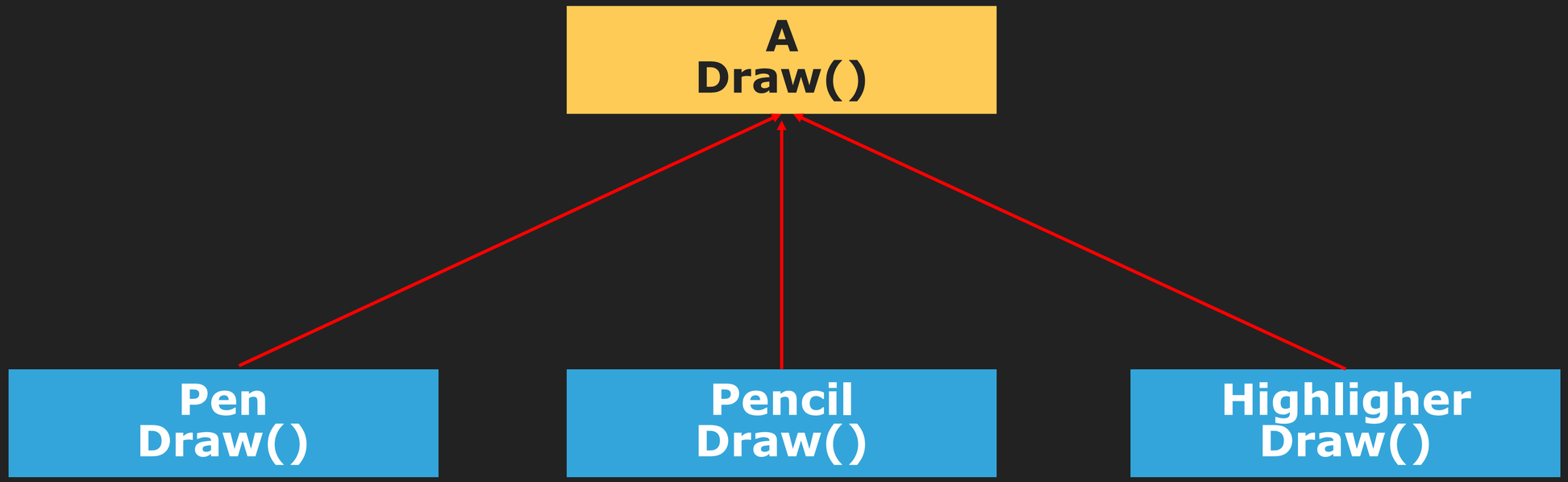
```
200
1100
```

Q6* : Explain "Overload" and "Override"
Q7* : Explain the difference between "Abstract Class" and "Interface"

Interface

Polymorphism

- Being able to be in many forms by inheritance



P8 : Compose a program as above

P9(Cont’d)

Course Registration Program

- Menu

1. Course List
2. My Course

-> 1

- Course List

1. Korean
2. Math
3. English
4. Social Studies
5. Science
0. Back to Main

-> 4

“Social Studies” is registered.

P9(Cont’d)

Course Registration Program

-> 4
“Social Studies” is registered.

▸ Course List
1. Korean
2. Math
3. English
4. Social Studies
5. Science
0. Back to Main
-> 0

▸ Menu
1. Course List
2. My Course
-> 2

P9

Course Registration Program

2. My Course
-> 2

▸ My Course
1. Social Studies
- End –
-> 0

▸ Menu
1. Course List
2. My Course
-> 1

P10

Reverse a Number

- User inputs a number
- Print the reversed number
- Use a method and parameter

245823
328542

P11

Reverse a String

- User inputs a string
- Print the reversed string
- Use a method and parameter

- Hint, “charAt()”

Hello
olleH

P12

Finding the Nearest Number

- User inputs a series of numbers
- (ex, 1.1 2.2 -5.7 10.3 20.2, -31.2)

- User inputs a number among the numbers above
  (ex, -5.7)

- Print the number which is the nearest one to the input number

1.1 2.2 -5.7 10.3 20.2 -31.2
-5.7
2.2

P13

Unit Converter

- Print the menu

```
# Unit
1. cm
2. m
3. mm
4. km
5. mile
```

- User selects the two units and inputs a number

- Print the result

- Use at least one class and one method

```
3 1
33
3.3
```