

Object – Oriented Programming

LAB #3. CLASS 1

Class

- 객체들을 여러 개 만들기 위한 하나의 틀
- 객체(Object) = 객체 지향 기술의 핵심 개념
= Field, method와 constructor로 이루어져 있다.
- 일반적으로 Class 이름의 첫 글자는 대문자로 한다.

```
public class Student{  
    ...  
}
```

Class

클래스 이름

```
public class Student {
```

Field

```
    private String name;  
    private String major;  
    private int grade;
```

Constructor

```
    public Student() {  
        ...  
    }  
    public Student(String _name, String _major, int _grade) {  
        ...  
    }
```

Method

```
    public void showInfo() {  
        ...  
    }
```

```
}
```

Class example – Date(1)

```
public class Date {  
    public String month;  
    public int day;  
    public int year;  
  
    public Date(String m, int d, int y){  
        month = m;  
        day = d;  
        year = y;  
    }  
    ...  
}
```

The new Operator

- 클래스의 객체를 생성하는데 사용하는 키워드
- 클래스 객체를 변수로 만들어 사용할 수 있다.

```
Date date1 = new Date();
```

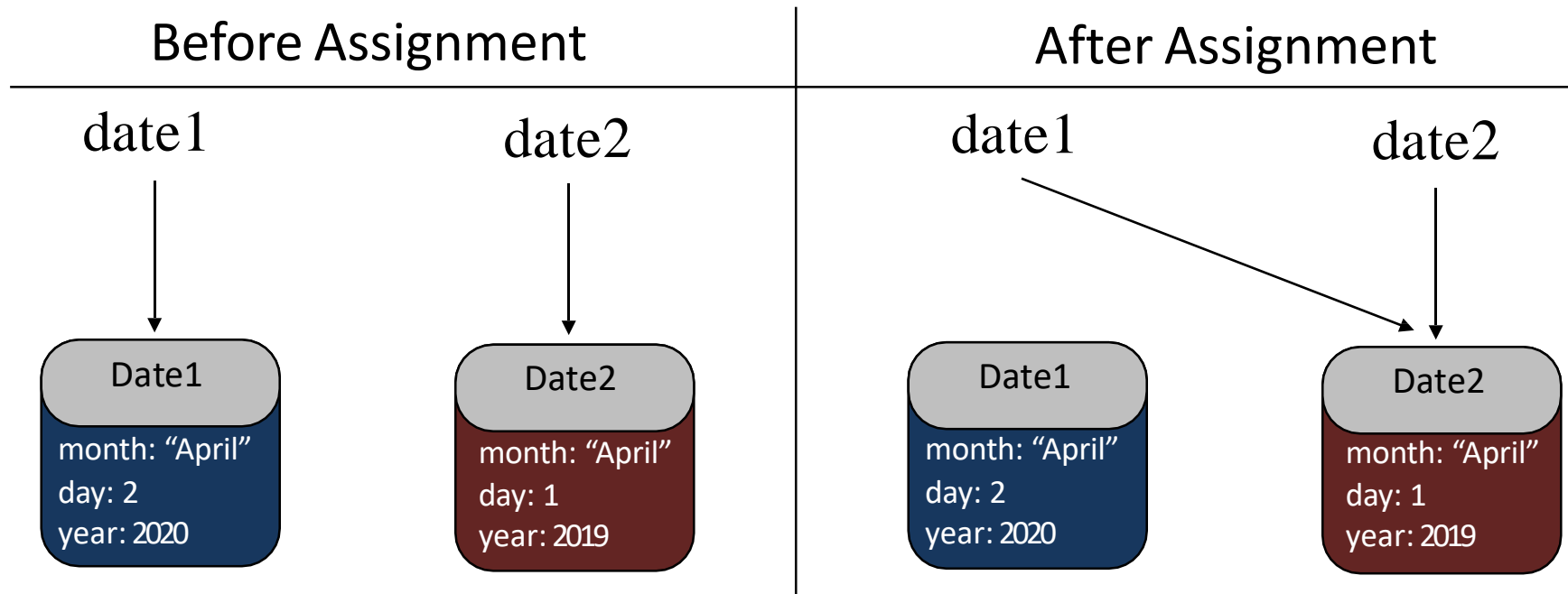
Creating objects of a class

```
Date date1;
```

```
date1 = new Date("April", 2, 2020);
```

```
Date date2 = new Date("April", 1, 2019);
```

```
date1 = date2;
```



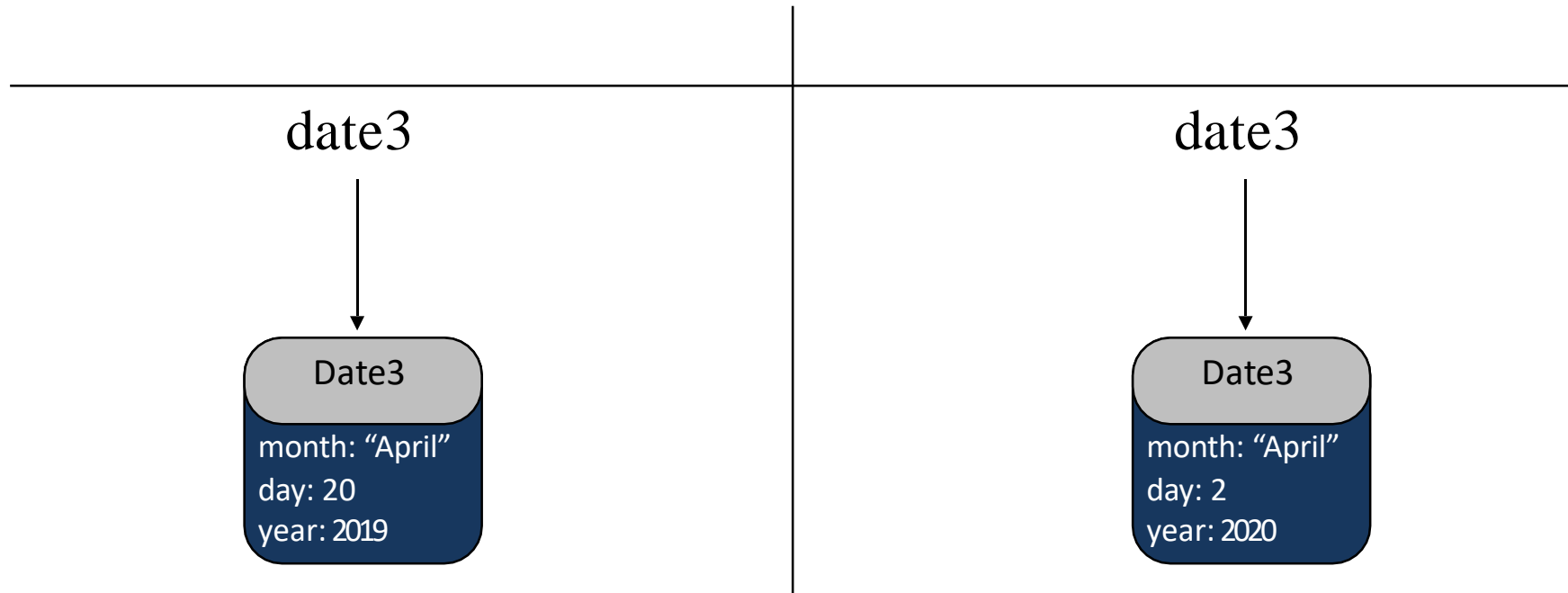
Instance Variables

```
Date date3;
```

```
date3 = new Date("April", 20, 2019);
```

```
date3.day = 2;
```

```
date3.year = 2020;
```



Method

- 클래스 내에 있는 함수

데이터 반환형 타입

메소드 선언

```
public void daysPassed(int n){  
    System.out.println(month + " " + (day + n) + ", " + year);  
}
```

메소드 호출

```
Date date4 = new Date("April", 2, 2020);  
date4.daysPassed(3);
```


Class example – Date(2)

```
public class Date {  
    public String month;  
    public int day;  
    public int year;  
  
    public Date(String m, int d, int y){  
        month = m;  
        day = d;  
        year = y;  
    }  
  
    public void daysPassed(int n){  
        System.out.println(month + " " + (day + n) + ", " + year);  
    }  
    ...  
}
```

The return Operator

- Method의 실행을 즉시 중지시키는 키워드
- 값을 반환하며 해당 Method를 중지시킨다.
- void 형식은 값을 반환하지 않는다.

```
public static void foo (int x) {  
    if ( x == 1 )  
    {  
        return;  
    }  
    System.out.println ("x is not 1");  
}
```

- void 형식이므로 return하면서 값을 반환하지 않는다.

this

- 현재 소속되어 있는 클래스를 가리키는 키워드
- Method 의 parameter와 Class 의 instance variable이 같은 변수명을 갖고 있다면, Class 의 instance variable를 가리킨다.

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

Class example – Date(3)

```
public class Date {  
    public String month;  
    public int day;  
    public int year;  
  
    public Date(String month, int day, int year){  
        this.month = month;  
        this.day = day;  
        this.year = year;  
    }  
  
    public void daysPassed(int n){  
        System.out.println(this.month + " " + (this.day + n) + ", " + this.year);  
    }  
    ...  
}
```

Concepts

- **Information hiding**: 클래스의 사용 방법과 그 구현의 세부사항을 구분하는 개념
 - Abstraction : 정보 과부하를 피하기 위해 세부사항을 폐기하는 개념
- **Encapsulation**: data와 Method를 클래스 내부에 결합시켜 구현 세부사항을 숨기는 개념
 - 개체와의 상호작용은 잘 정의되고 단순한 인터페이스를 통해 이루어지므로 세부 정보를 알 필요가 없다.
 - 자바에서는 세부 정보를 Private으로 하여 숨긴다.

Modifiers

- public: 어디에서든 접근 가능
- private: 해당 클래스에서만 접근 가능

Class example – Date(4)

```
public class Date {  
    private String month;  
    private int day;  
    private int year;  
  
    public Date(String month, int day, int year){  
        this.month = month;  
        this.day = day;  
        this.year = year;  
    }  
  
    public void daysPassed(int n){  
        System.out.println(this.month + " " + (this.day + n) + ", " + this.year);  
    }  
    ...  
}
```

Accessor-Mutator Method

- Accessor method (getter) : Private 접근 한정자를 지닌 값을 반환하는 Method
Method 이름의 시작 부분에 'get'이라는 접두사를 붙인다.
- Mutator method (setter) : Private 접근 한정자를 지닌 값을 설정하는 Method
Method 이름의 시작 부분에 'set'이라는 접두사를 붙인다.

ex)

```
private String address = 10;  
  
public String getAddress() {  
    return address;  
}
```

```
public void setAddress(String address) {  
    this.address = address;  
}
```


Class example – Date(5)

```
public class Date {  
    private String month;  
    private int day;  
    private int year;  
  
    ...  
  
    public int getDay(){  
        return day;  
    }  
  
    public void setDay(int day){  
        this.day = day;  
    }  
  
    ...  
}
```

Method Overloading

- 모든 Method는 각자 다른 Signature를 가져야한다.
- Method는 매개 변수가 다를 경우 동일한 이름을 가질 수 있다.

```
package practice03_1;

public class Practice03_1 {

    public void add(int a, int b) {
        System.out.println("더한 결과는 (" + (a + b) + ")입니다.");
    }

    public double add(double a, double b) {
        return a + b;
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Practice03_1 test = new Practice03_1();

        test.add(10, 20);
        test.add(10.5, 20.4);

        System.out.println(test.add(10.5, 20.4));
    }
}
```

Method Overloading

```
package practice03_1;

public class Practice03_1 {

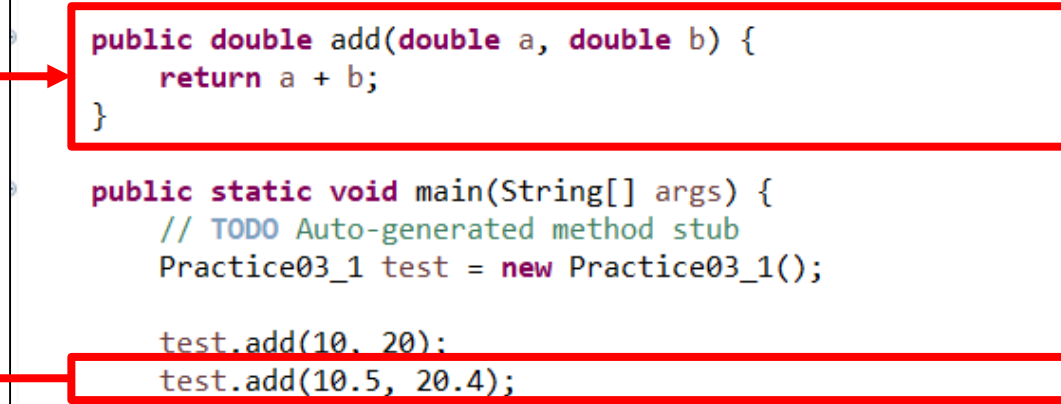
    public void add(int a, int b) {
        System.out.println("더한 결과는 (" + (a + b) + ")입니다.");
    }

    public double add(double a, double b) {
        return a + b;
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Practice03_1 test = new Practice03_1();

        test.add(10, 20);
        test.add(10.5, 20.4);

        System.out.println(test.add(10.5, 20.4));
    }
}
```



Constructor

- 클래스를 생성할 때 한번만 호출되는 메소드
- 일반적으로 변수를 초기화하는데 사용된다.
- “default” 생성자는 매개변수가 없다.

클래스 이름

```
public Date() {  
    ...  
}  
public Date(String month, int day, int year) {  
    ...  
}
```

접근 제한자 ←

매개 변수 = 메소드를 호출할 때 값을 넘겨 받음

Class example – Date(6)

```
public class Date {  
    ...  
    public Date(){  
        month = "April";  
        day = 2;  
        year = 2020;  
    }  
  
    public Date(String month, int day, int year){  
        this.month = month;  
        this.day = day;  
        this.year = year;  
    }  
    ...  
}
```

equals()

- JAVA는 개체가 다른 개체와 동일한지에 대한 여부를 확인할 수 있다.
- equals() Method는 Boolean 값을 return 한다. (동일하면 true / 아니면 false)

```
public boolean equals(Person anotherPerson){  
    if (this.name.equals(anotherPerson.name) && this.age == anotherPerson.age)  
        return true;  
    else  
        return false;  
}
```

toString()

- JAVA에서 개체는 문자열을 반환하는 toString() Method를 가진다.
- 프로그래머는 toString() method 가 반환하는 정보를 정할 수 있다.

```
public String toString(){  
    return "Person:: \n"+  
        "Name: "+this.name+"\n"+  
        "Age: "+this.age+"\n";  
}
```

Class example – Date(7)

```
public class Date {  
    ...  
    public Boolean equals(Date anotherDate){  
        if(this.month.equals(anotherDate.month)  
            && this.day == anotherDate.day)  
            return true;  
        else  
            return false;  
    }  
  
    public String toString(){  
        return month + " " + day + ", " + year;  
    }  
    ...  
}
```


Class 실습 과제

Employee.class

- Employee 라는 이름을 가진 class를 만든다.
- 클래스는 다음과 같은 정보를 담는다.
 - Instance variables: name(String), age(int), position(String), salary(int), vacationDays(int) (모두 private)
 - 3개의 생성자
 - (1) name과 age만을 set하는 생성자 – position: “Employee”, salary: 5,000, vacationDays: 20으로 기본 설정
 - (2) name, age, position, salary만을 set하는 생성자 – vacationDays를 기본적으로 20일로 설정
 - (3) 모든 변수를 set하는 생성자
 - 각 변수들을 설정하고 반환하는 getter, setter Method (public)
 - 직원이 동일한지 판단하는 public equals method (name, age, position 이 같으면 같은 것으로 판단)
 - 직원의 정보를 출력하는 public toString method
 - ex) Name : James Wright, Age : 42, Position : Manager, Salary : 20000, VacationDays: 20
 - vacation이라는 이름을 가진 Method (public boolean)
 - 직원이 휴가를 내고 싶을 때, 사용하는 Method.
 - 호출 시, 매개변수로 사용하고 싶은 휴가 일 수를 입력한다.
 - 현재 vacationDays의 값보다 많은 휴가일을 사용할 경우, vacationDays를 변경하지 않고 “남은 휴가 일수가 부족합니다.” 라는 메시지 출력하고 false를 반환한다.
 - 휴가일이 남아있을 경우, vacationDays를 휴가 일 수만큼 감소시키고 “휴가를 사용하였습니다. 남은 휴가 일 수 : %d” 라는 메시지 출력한다. %d 에는 vacationDays를 출력하고 true를 반환한다.

Class 실습 과제

EmployeeManager.class

- EmployeeManager 라는 이름을 가진 Application을 생성하고 main Method에서 다음 동작 수행
- 다음과 같은 정보를 가진 employees를 추가
 - Name: James Wright, Age: 42, Position: Manager, Salary: 20000
 - Name: Amy Smith, Age: 27, Position: Design Coordinator, Salary: 8000 , vacationDays: 15
 - Name: Peter Coolidge, Age: 32, Position: Assistant Manager, Salary: 12000, vacationDays: 7
 - Name: John Doe, Age: 22, Position: Engineer, Salary: 10000, vacationDays: 10
- 직원을 추가할 때, 다음과 같이 직원의 정보를 출력한다. (toString Method 사용)
 - Name: Amy Smith, Age: 27, Position: Design Coordinator, Salary: 18000
- 새로운 employee를 생성하고 Amy 와 비교한 결과값을 출력한다. (equals Method 사용)
- James 객체가 10일 휴가를 사용하게 한다.
- Peter 객체가 10일 휴가를 사용하게 한다.
- 모든 직원들의 정보를 출력한다.

Class 실습 과제

Employee.java와 EmployeeManager.java를 Blackboard에 제출

```
Name: James Write, Age: 42, Postion: Manager, Salary: 20000, VacationDays: 20
Name: Amy Smith, Age: 27, Postion: Design Coordinator, Salary: 8000, VacationDays: 15
Name: Peter Coolidge, Age: 32, Postion: Assistant Manager, Salary: 12000, VacationDays: 7
Name: John Doe, Age: 22, Postion: Engineer, Salary: 10000, VacationDays: 10
휴가를 사용하였습니다. 남은 휴가 일 수 : 10
남은 휴가 일수가 부족합니다.
Name: James Write, Age: 42, Postion: Manager, Salary: 20000, VacationDays: 10
Name: Amy Smith, Age: 27, Postion: Design Coordinator, Salary: 8000, VacationDays: 15
Name: Peter Coolidge, Age: 32, Postion: Assistant Manager, Salary: 12000, VacationDays: 7
Name: John Doe, Age: 22, Postion: Engineer, Salary: 10000, VacationDays: 10
```