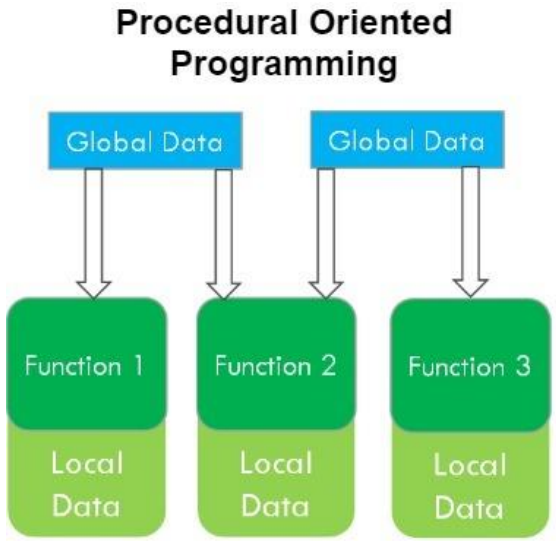
객체지향 프로그래밍 및 실습

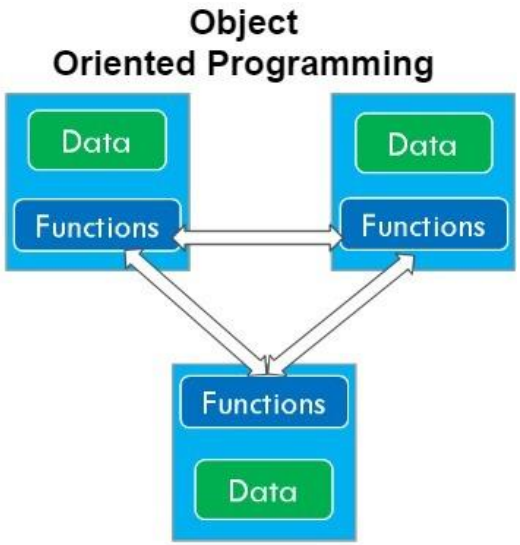
3주차. 단순한 Classes, Objects, Methods

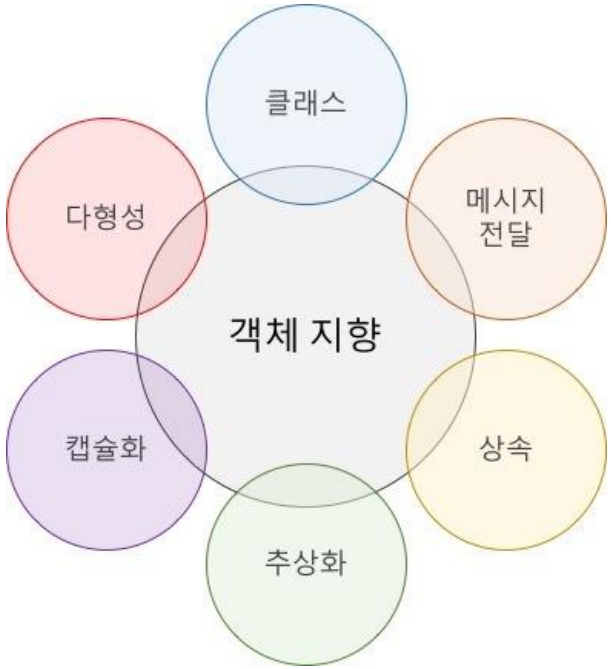
1. 클래스와 객체

1. 클래스와 객체

- Class와 Object
  - 객체 지향 프로그래밍(Object Oriented Programming)에서 Class와 Object에 대한 개념은 매우 중요
  - 프로그램을 명령어의 집합(ex. Procedure)에서 벗어나 객체들의 모임으로 보고자 하는 시각 → 객체지향







1. 클래스와 객체

- Class와 Object
  - Class는 단순 **설계도**
  - Class를 실체화(instantiation) 하는 것 → 객체(Object) 또는 인스턴스


```java
public class Car {
    String manufacturer;    // 제조사
    String modelName;  // 자동차 모델명
    int horsePower; // 마력
    int maxSpeed;   // 최대 속도
    int currentSpeed;   // 현재 속도

    public Car(String manufacturer, String modelName, int horsePower, int maxSpeed) {
        this.manufacturer = manufacturer;
        this.modelName = modelName;
        this.horsePower = horsePower;
        this.maxSpeed = maxSpeed;
        this.currentSpeed = 0;
    }

    /**
     * 악셀 밟기
     */
    public void pushGasPedal() {
        currentSpeed += horsePower;
    }
}
```

1. 클래스와 객체

- Class와 Object
  - Car이라고 하는 클래스를 기반으로 Audi, Nissan, Volvo 객체를 생성할 수 있다.

  - Car type의 객체는 같은 이름의 행동(Method)를 할 수 있고, 같은 이름의 속성(Attribute)를 가진다.
  - 하지만 행동, 속성의 세부사항 및 동작은 각자 다르다.

```
1 Car q3 = new Car("Audi", "Q3", 183, 207);
2
3 Car ioniq5 = new Car("Hyundai", "IONIQ3", 235, 185);
4
5 Car morning = new Car("Kia", "All new Morning", 84, 160);
```

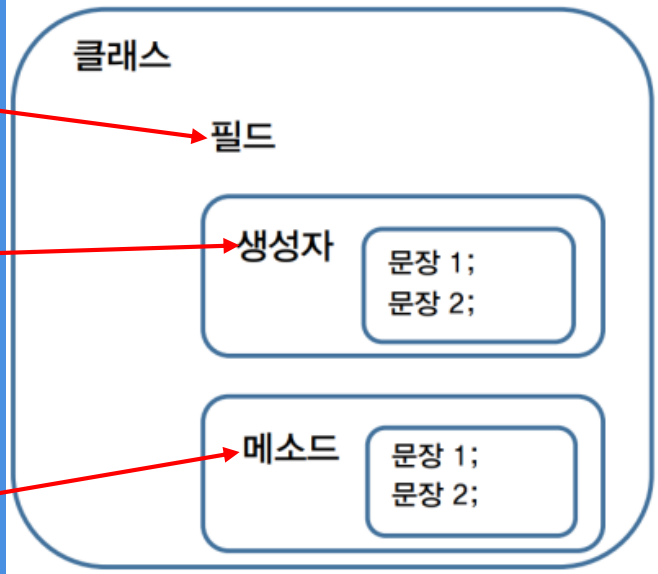
1. 클래스와 객체

- Class와 Object

```java
public class Car {
    String manufacturer;    // 제조사
    String modelName;  // 자동차 모델명
    int horsePower; // 마력
    int maxSpeed;   // 최대 속도
    int currentSpeed;   // 현재 속도

    public Car(String manufacturer, String modelName, int horsePower, int maxSpeed) {
        this.manufacturer = manufacturer;
        this.modelName = modelName;
        this.horsePower = horsePower;
        this.maxSpeed = maxSpeed;
        this.currentSpeed = 0;
    }

    /**
     * 악셀 밟기
     */
    public void pushGasPedal() {
        currentSpeed += horsePower;
    }
}
```



1. 클래스와 객체

- **실습 문제1**
  - 학생 클래스를 Java 코드로 표현해 보자.
  - 학생 클래스 – 속성
    - 학생은 이름을 가진다
    - 학생은 학번을 가진다
    - 학생은 학년을 가진다
    - 학생은 학과를 가진다.
  - 학생 클래스 – 행동
    - 자기소개를 할 수 있다.
    - 졸업을 할 수 있다. - 4학년이 아니면 졸업할 수 없다.
    - 학교 내부를 돌아다닐 수 있다.

  - 답이 정확히 없는 문제이므로 각자 설계한 클래스 구조가 다를 수 있다.

2. 객체의 레퍼런스

2. 객체의 레퍼런스

- 객체의 생성과 참조(reference)
  - Class는 User-defined data type이다.
  - 따라서 Car의 인스턴스는 Car type이라고 할 수 있다.
  - 레퍼런스(reference, 또는 참조값)는 객체가 저장되어 있는 일종의 주소 값과 비슷

```
Car morning = new Car("Kia", "All new Morning", 84, 160);
```

타입 참조변수 레퍼런스

  - 위의 코드에서는 참조변수 morning에 '기아 올뉴모닝' 객체의 레퍼런스가 담긴다
  - 따라서 아래와 같은 코드도 가능

```
1 Car morning = new Car("Kia", "All new Morning", 84, 160);
2 Car morning2 = morning;
3
4 System.out.println(morning == morning2) // true
```

  - 이럴 경우, morning과 morning2는 둘다 같은 객체를 가리키게 된다.

2. 객체의 레퍼런스

- 객체의 접근 방법
  - 객체의 필드와 메소드는 '.'(Dot 연산자)를 통해서 접근 가능하다.

```java
public class Person {
    String name;    // 이름
    int age;    // 나이
    int height; // 키

    public void selfIntroduce() {
        System.out.printf("안녕하세요! 저는 %s입니다. 나이는 %d이고 키는 %d입니다.", name, age, height);
    }
}
```

```java
Person person1 = new Person();

person1.name = "김국디";    // person1이 참조하는 객체의 변수 name에 "김국디" 저장
person1.age = 21;          // ''' age에 21 저장
person1.height = 180;      // ''' height에 180 저장

person1.selfIntroduce();  // person1이 참조하는 객체의 메소드 selfIntroduce 호출
```

2. 객체의 레퍼런스

- String 클래스
  - String은 primitive type이 아닌, 클래스다.

```
1 public class String {
2   char[] value;
3
4   ...
5 }
```

  - ↑ String 클래스의 간략한 내부구조(value 배열에 문자열이 담김)

```
1 String a = new String("Hello");  // String a = "Hello"와 같은 동작
2 String b = new String("Hello");
3
4 System.out.println(a == b)  // true? or false?
```

  - 위의 코드에서 a와 b는 같을까?
  - 일단 두 인스턴스의 value 배열에는 같은 값이 담겨있을 것 같다.

2. 객체의 레퍼런스

- String 클래스

```
1 String a = new String("Hello");  // String a = "Hello"와 같은 동작
2 String b = new String("Hello");
3
4 System.out.println(a == b)  // true? or false?
```

  - a와 b에는 같은 값이 담겨있다.
  - 하지만 같은 객체는 아니다.

  - 따라서 a == b의 결과는 false!

  - 그러면 String 끼리는 어떻게 비교할까?
    - a.equals(b), b.equals(a)

2. 객체의 레퍼런스

- **실습 문제 2**
  - 다음 코드의 실행 예상 결과와 그 이유를 PersonTest.java파일 위에 주석으로 작성하시오.

```java
public class PersonTest {
    public static void main(String[] args) {
        Person a = new Person();
        Person b = new Person();
        Person c = a;

        a.age = 21;
        b.age = 22;
        c.age = 23;

        System.out.println("a: age is " + a.age);
        System.out.println("b: age is " + b.age);
        System.out.println("c: age is " + c.age);
    }
}
```

3. getter와 setter

3. getter와 setter

- 객체의 필드에 다이렉트로 접근하기

```
1 Person a = new Person();
2
3 a.age = 25;
4 System.out.println(a.age)
```

  - <참조변수>.<필드명>으로 다이렉트하게 접근할 수 있다.

```
1 Person a = new Person();
2
3 a.age = -24;
4 System.out.println(a.age)
```

  - 음수의 나이는 없다. 그렇다면 3번 라인은 명백한 오류!
  - 객체의 필드 값에 제약사항을 부여할 수는 없을까?
  - 외부에서 age변수가 보이지 않도록 해야겠다!

3. getter와 setter

- 필드 변수의 은닉화

```
1 public class Person {
2     private String name;     // 이름
3     private int age;     // 나이
4     private int height; // 키
5 }
```

  - 필드 변수의 앞에 private를 넣어주었다.

```
1 public class PersonTest {
2     public static void main(String[] args)
3         Person a = new Person();
4
5         a.age = 22;
6     }
7 }
```

  - Person 클래스의 외부에서는 Person.name, Person.age, Person.height가 안보이게 되는 효과가 발생한다.

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        The field Person.age is not visible

        at PersonTest.main(PersonTest.java:5)
```

3. getter와 setter

- getter와 setter
  - Person클래스의 외부에서는 name, age, height가 안보이게 됐다. 그렇다면 어떻게 접근해야 할까?
  - Setter를 사용하여 접근
  - 필요에 따라서 제약을 걸 수 있음

```java
public class Person {
    private String name;
    private int age;
    private int height;

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

    public void setAge(int age) {
        if (age > 0) {
            this.age = age;
        } else {
            System.out.println("age는 음수가 될 수 없습니다!");
        }
    }

    public void setHeight(int height) {
        if (height > 0) {
            this.height = height;
        } else {
            System.out.println("height는 음수가 될 수 없습니다!");
        }
    }
}
```

3. getter와 setter

- getter와 setter
  - Person클래스의 외부에서는 name, age, height가 안보이게 됐다. 그렇다면 어떻게 접근해야 할까?
  - Getter를 사용하여 접근

```java
public class Person {
    private String name;
    private int age;
    private int height;

    public String getName() {
        return this.name;
    }

    public int getAge() {
        return this.age;
    }

    public int getHeight() {
        return this.height;
    }
}
```

3. getter와 setter

- **실습 문제 3**
  - 정다각형의 내각의 합, 한 내각의 크기, 한 외각의 크기를 구하기 위한 프로그램을 만들어 보자!
  - https://shorturl.at/gsvDT ← 빈칸에 들어갈 코드를 채워 보기

```java
public class RegularPolygonTest {
    public static void main(String[] args) {
        RegularPolygon triangle = new RegularPolygon(3);
        RegularPolygon square = new RegularPolygon(4);

        System.out.printf("삼각형의 내각의 합: %d\n", triangle.getSumOfInternalAngles());
        System.out.printf("삼각형의 한 내각의 크기: %d\n", triangle.getInternalAngle());
        System.out.printf("삼각형의 한 외각의 크기: %d\n", triangle.getExteriorAngle());
        System.out.println();
        System.out.printf("사각형의 내각의 합: %d\n", square.getSumOfInternalAngles());
        System.out.printf("사각형의 한 내각의 크기: %d\n", square.getInternalAngle());
        System.out.printf("사각형의 한 외각의 크기: %d\n", square.getExteriorAngle());
    }
}
```

3. getter와 setter

- **실습 문제 3**
  - 참고사항 – 정n각형
    - 내각의 크기의 합: 180 * (n-2)
    - 한 내각의 크기: (내각의 크기의 합)/n
    - 외각의 크기: 360
    - 한 외각의 크기: (외각의 크기) / n

3. getter와 setter

- Immutable(불변) 객체
  - 실습 문제 3의 RegularPolygon클래스는 내부의 객체 생성 후, 내부의 상태를 바꿀 수 없다.
    - numOfAngles 변수에 대한 setter가 없고, 외부에서도 접근할 수 없기 때문
    - 따라서 삼각형(triangle 참조변수)은 끝까지 삼각형! 삼각형을 사각형으로 바꿀 수 없다
  - 이러한 객체를 Immutable 객체라고 한다.

  - Immutable 클래스의 대표적인 예시
    - String, Boolean, Integer, Long

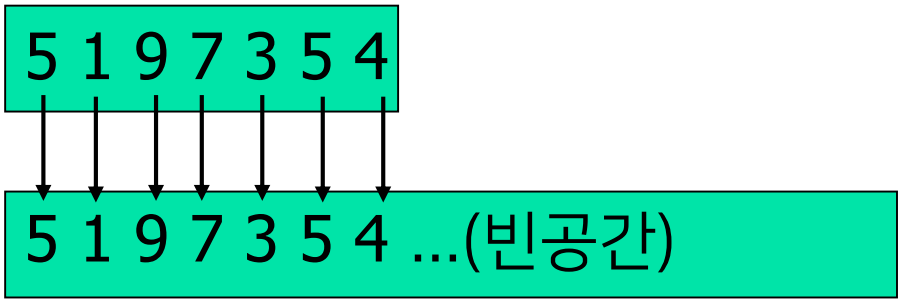
```
1 String a = "HI";
2 a += "Hello";
3 a += "Whatup";
4 System.out.println(a);
```

    - a의 참조객체에 Hello, Whatup 문자열이 추가되는 방식이 아니다.(immutable하기 때문)
    - “HI”, “HIHello”, “HIHelloWhatup” 객체가 총 3개나 생성되었다.

4. ArrayList

4. ArrayList

- ArrayList의 동작 원리



  - 배열이 꽉 찼다면, 길이가 2배인 배열을 새로 만들고 여기에 기존 값들을 복사한다
  - 사용자 입장에서는 길이가 무제한에 가까운 배열!

4. ArrayList

- ArrayList의 사용 방법
  - https://shorturl.at/fkvyE

```java
import java.util.ArrayList;
import java.util.List;

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

        // List<type> name = new ArrayList<>(); 방식으로 생성한다.
        List<String> strList = new ArrayList<>();

        strList.add("HI");
        strList.add("Hello");
        strList.add("Whatup");

        System.out.println(strList.size()); // get size(length) of list
        System.out.println(strList.isEmpty()); // same of 'strList.size() == 0'
        System.out.println(strList.get(2)); // 2번째 element를 리스트에서 읽어옴


        // foreach 문으로도 활용가능
        for (String value : strList) {
            System.out.println(value);
        }

        strList.remove(2); // 2번째 element를 리스트에서 삭제
        strList.remove("HI");   // HI라는 값을 가진 element를 리스트에서 삭제
    }
}
```

4. ArrayList

- **실습 문제 4** – Coin.java
  - Coin을 평가 금액을 반환하는 calcPrice() 메소드 작성
  - 생성자 작성
  - Getter, Setter 작성

```java
public class Coin {
    private String coinName;    // 코인 이름
    private double price;       // 가격
    private int qty;            // 보유 개수

    // TODO 생성자를 작성하세요

    /**
     * 평가 금액 (가격 * 보유 개수)
     */
    public int calcPrice() {
        // TODO 메소드를 채워주세요
    }

    @Override
    public String toString() {
        return String.format("코인이름: %s  가격: %d  보유수량: %d  평가 금액: %d", coinName, price, qty,
calcPrice());
    }
}
```

4. ArrayList

- **실습 문제 4** – CoinTest.java
  - Coin을 담는 ArrayList - myWallet를 생성
  - myWallet에 아래 사항을 담도록 구성

이름	가격	개수
BTC	9543.66	10
ETH	3240.17	20
XRP	910.95	10
LTC	1184.12	30

  - ForEach문을 통해 리스트에 담긴 각각의 Coin.toString()을 호출 -> 이를 출력

<출력 예시>
코인이름: BTC 가격: 9543.66 보유수량: 10 평가 금액: 95436.6
코인이름: ETH 가격: 3240.17 보유수량: 20 평가 금액: 64803.4
코인이름: XRP 가격: 910.95 보유수량: 10 평가 금액: 9109.5
코인이름: LTC 가격: 1184.12 보유수량: 30 평가 금액: 35523.6