

Chapter 7

객체지향 프로그래밍 II
Object-oriented Programming II

[연습문제]

[7-1] 섯다카드 20장을 포함하는 섯다카드 한 벌(SutdaDeck클래스)을 정의한 것이다. 섯다카드 20장을 담은 SutdaCard배열을 초기화하시오. 단, 섯다카드는 1부터 10까지의 숫자가 적힌 카드가 한 쌍씩 있고, 숫자가 1, 3, 8인 경우에는 둘 중의 한 장은 광(Kwang)이어야 한다. 즉, SutdaCard의 인스턴스변수 isKwang의 값이 true이어야 한다.

[연습문제]/ch7/Exercise7_1.java

```
class SutdaDeck {
    final int CARD_NUM = 20;
    SutdaCard[] cards = new SutdaCard[CARD_NUM];

    SutdaDeck() {
        /*
            (1) 배열 SutdaCard를 적절히 초기화 하시오.
        */
        for (int i=0 ; i<cards.length ; i++) {
            int num = i % 10+1;
            boolean isKwang = (i<10)&&(num == 1 || num == 3 || num == 8);
            cards[ i ] = new SutdaCard(num, isKwang);
        }
    }

    class SutdaCard {
        int num;
        boolean isKwang;

        SutdaCard() {
            this(1, true);
        }

        SutdaCard(int num, boolean isKwang) {
            this.num = num;
            this.isKwang = isKwang;
        }

        // info() 대신 Object클래스의 toString()을 오버라이딩했다.
        public String toString() {
            return num + ( isKwang ? "K":"" );
        }
    }

    class Exercise7_1 {
        public static void main(String args[]) {
            SutdaDeck deck = new SutdaDeck();

            for(int i=0; i < deck.cards.length;i++)
                System.out.print(deck.cards[i]+" ");
        }
    }
}
```

i	i%10	i%10+1
0	0	1
1	1	2
2	2	3
3	3	4
4	4	5
5	5	6
6	6	7
7	7	8
8	8	9
9	9	10
10	0	1
11	1	2
12	2	3
13	3	4
14	4	5
15	5	6
16	6	7
17	7	8
18	8	9
19	9	10

[실행결과]

```
1K,2,3K,4,5,6,7,8K,9,10,1,2,3,4,5,6,7,8,9,10,
```

[7-2] 문제7-1의 SutdaDeck클래스에 다음에 정의된 새로운 메서드를 추가하고 테스트 하시오.

[주의] Math.random()을 사용하는 경우 실행결과와 다를 수 있음.

1. 메서드명 : shuffle

기 능 : 배열 cards에 담긴 카드의 위치를 뒤섞는다.(Math.random()사용)

반환타입 : 없음

매개변수 : 없음

2. 메서드명 : pick

기 능 : 배열 cards에서 지정된 위치의 SutdaCard를 반환한다.

반환타입 : SutdaCard

매개변수 : int index - 위치

3. 메서드명 : pick

기 능 : 배열 cards에서 임의의 위치의 SutdaCard를 반환한다.(Math.random()사용)

반환타입 : SutdaCard

매개변수 : 없음

[연습문제]/ch7/Exercise7_2.java

```
class SutdaDeck {
    final int CARD_NUM = 20;
    SutdaCard[] cards = new SutdaCard[CARD_NUM];

    SutdaDeck() {
        /*
            문제 7-1의 답이므로 내용생략
        */
    }

    /*
        (1) 위에 정의된 세 개의 메서드를 작성하시오.
    */

} // SutdaDeck

class SutdaCard {
    int num;
    boolean isKwang;

    SutdaCard() {
        this(1, true);
    }

    SutdaCard(int num, boolean isKwang) {
        this.num = num;
        this.isKwang = isKwang;
    }

    public String toString() {

void shuffle() {
    for (int i=0 ; i<cards.length ; i++) {
        int j = Math.random()*cards.length;

        SutdaCard tmp = cards[ i ];
        cards[ i ] = cards[ j ];
        cards[ j ] = tmp;
    }
}

SutdaCard pick(int index) {
    if (index < 0 || index >= CARD_NUM) {
        return null;
    }
    return cards[index];
}

SutdaCard pick() {
    int index = Math.random()*cards.length;
    return pick(index);
}

}
```

```

        return num + ( isKwang ? "K":"" );
    }
}

class Exercise7_2 {
    public static void main(String args[]) {
        SutdaDeck deck = new SutdaDeck();

        System.out.println(deck.pick(0));
        System.out.println(deck.pick());
        deck.shuffle();

        for(int i=0; i < deck.cards.length;i++)
            System.out.print(deck.cards[i]+",");

        System.out.println();
        System.out.println(deck.pick(0));
    }
}

```

[실행결과]

```

1K
7
2,6,10,1K,7,3,10,5,7,8,5,1,2,9,6,9,4,8K,4,3K,
2

```


[7-3] 오버라이딩의 정의와 필요성에 대해서 설명하시오.

오버라이딩이란 상속 관계에 있는 부모 클래스에서 이미 정의된 메소드를 자식 클래스에서 같은 시그니처를 갖는 메소드로 다시 정의하는 것이다.

부모 클래스로부터 상속받은 메소드를 자식 클래스에서 그대로 사용할 수 없는 경우가 많기 때문에 오버라이딩이 필요하다.

[7-4] 다음 중 오버라이딩의 조건으로 옳지 않은 것은? (모두 고르시오)

- 조상의 메서드와 이름이 같아야 한다. **이름, 매개변수, 리턴타입이 같아야 한다.**
- 매개변수의 수와 타입이 모두 같아야 한다.

 접근 제어자는 조상의 메서드보다 좁은 범위로만 변경할 수 있다.

 조상의 메서드보다 더 많은 수의 예외를 선언할 수 있다.

+ 인스턴스 메소드를 static 메소드로 또는 그 반대로 변경할 수 없다.

[7-5] 다음의 코드는 컴파일하면 에러가 발생한다. 그 이유를 설명하고 에러를 수정하기 위해서는 코드를 어떻게 바꾸어야 하는가?

[연습문제]/ch7/Exercise7_5.java

```
class Product
{
    int price;          // 제품의 가격
    int bonusPoint;     // 제품구매 시 제공하는 보너스점수

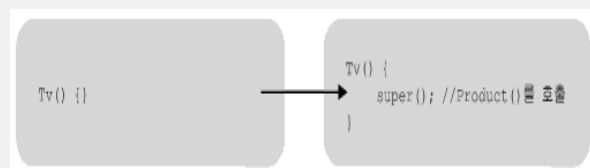
    Product(int price) {
        this.price = price;
        bonusPoint = (int) (price/10.0);
    }
}

class Tv extends Product {
    Tv() {}

    public String toString() {
        return "Tv";
    }
}

class Exercise7_5 {
    public static void main(String[] args) {
        Tv t = new Tv();
    }
}
```

Product() {}



컴파일러가 자동 추가

기본 생성자 추가

실제 코드에서는 `super()`를 호출하는 곳이 없지만, 컴파일러가 자동으로 추가해 준다. `super()`에서 `Product()`를 호출하지만, 정의되어 있지 않은 생성자를 호출하기 때문에 에러가 발생한다.

[7-6] 자손 클래스의 생성자에서 조상 클래스의 생성자를 호출해야하는 이유는 무엇인가?

부모 클래스에 정의된 인스턴스 변수를 초기화 하기 위해서

각 클래스의 생성자는 해당 클래스에 선언된 인스턴스 변수의 초기화만 담당하고, 부모 클래스로부터 상속받은 인스턴스 변수의 초기화는 부모 클래스의 생성자가 처리하도록 해야 한다.

[7-7] 다음 코드의 실행했을 때 호출되는 생성자의 순서와 실행결과를 적으시오.

[연습문제]/ch7/Exercise7_7.java

```
class Parent {
    int x=100;

    Parent() {
        this(200);
    }

    Parent(int x) {
        this.x = x;
    }

    int getX() {
        return x;
    }
}

class Child extends Parent {
    int x = 3000;

    Child() {
        this(1000);
    }

    Child(int x) {
        this.x = x;
    }
}

class Exercise7_7 {
    public static void main(String[] args) {
        Child c = new Child();

        System.out.println("x="+c.getX());
    }
}
```

Child() > Child(int x) > Parent() > Parent(int x) > getX()

호출 순서에 따라 Child 클래스의 변수 x는 1000, Parent 클래스의 변수 x는 200 이다.
getX()는 부모인 Parent 클래스에서 정의된 것이기 때문에 getX()의 x는 Parent 클래스의 x를 의미한다.

[7-8] 다음 중 접근제어자를 접근범위가 넓은 것에서 좁은 것의 순으로 바르게 나열한 것은?

- ☒ a. public-protected-(default)-private
- b. public-(default)-protected-private
- c. (default)-public-protected-private
- d. private-protected-(default)-public

private : 같은 클래스 내에서만 접근 가능
default : 같은 패키지 내에서만 접근 가능
protected : 같은 패키지 내, 다른 패키지 자식 클래스에서 접근 가능
public : 접근 제한 없음

[7-9] 다음 중 제어자 final을 붙일 수 있는 대상과 붙였을 때 그 의미를 적은 것이다. 옳지 않은 것은? (모두 고르시오)

- a. 지역변수 - 값을 변경할 수 없다.
- b. 클래스 - 상속을 통해 클래스에 새로운 멤버를 추가할 수 없다.
- ☒ c. 메서드 - 오버로딩을 할 수 없다. 오버라이딩을 할 수 없다.
- d. 멤버변수 - 값을 변경할 수 없다.

제어자 final은 '마지막의' 또는 '변경될 수 없는'의 의미를 가지고 있으며 거의 모든 대상에 사용될 수 있다.

제어자	대상	의미
final	클래스	변경될 수 없는 클래스, 확장될 수 없는 클래스가 된다. 그래서 final로 지정된 클래스는 다른 클래스의 조상이 될 수 없다.
	메서드	변경될 수 없는 메서드, final로 지정된 메서드는 오버라이딩을 통해 재정의 될 수 없다.
	멤버변수 지역변수	변수 앞에 final이 붙으면, 값을 변경할 수 없는 상수가 된다.

[7-10] MyTv2클래스의 멤버변수 isPowerOn, channel, volume을 클래스 외부에서 접근할 수 없도록 제어자를 붙이고 대신 이 멤버변수들의 값을 어디서나 읽고 변경할 수 있도록 getter와 setter메서드를 추가하라.

[연습문제]/ch7/Exercise7_10.java

```
class MyTv2 {
    boolean isPowerOn;
    int channel;
    int volume;

    final int MAX_VOLUME = 100;
    final int MIN_VOLUME = 0;
    final int MAX_CHANNEL = 100;
    final int MIN_CHANNEL = 1;

    /*
        (1) 알맞은 코드를 넣어 완성하시오.
    */
}

class Exercise7_10 {
    public static void main(String args[]) {
        MyTv2 t = new MyTv2();

        t.setChannel(10);
        System.out.println("CH:"+t.getChannel());
        t.setVolume(20);
        System.out.println("VOL:"+t.getVolume());
    }
}
```

```
public void setVolume(int volume) {
    if (volume > MAX_VOLUME ||
        volume < MIN_VOLUME) {
        return;
    }
    this.volume = volume;
}

public int getVolume() {
    return volume;
}

public void setChannel(int channel) {
    if (channel > MAX_CHANNEL ||
        channel < MIN_CHANNEL) {
        return;
    }
    this.channel = channel;
}

public int getChannel() {
    return channel;
}
```

[실행결과]

```
CH:10
VOL:20
```

[7-11] 문제7-10에서 작성한 MyTv2클래스에 이전 채널(previous channel)로 이동하는 기능의 메서드를 추가해서 실행결과와 같은 결과를 얻도록 하시오.

[Hint] 이전 채널의 값을 저장할 멤버변수를 정의하라.

메서드명 : gotoPrevChannel
 기 능 : 현재 채널을 이전 채널로 변경한다.
 반환타입 : 없음
 매개변수 : 없음

[연습문제]/ch7/Exercise7_11.java

```
class MyTv2 {
    /*
        (1) 문제7-10의 MyTv2클래스에 gotoPrevChannel메서드를 추가하여 완성하시오.
    */
}

class Exercise7_11 {
    public static void main(String args[]) {
        MyTv2 t = new MyTv2();

        t.setChannel(10);
        System.out.println("CH:"+t.getChannel());
        t.setChannel(20);
        System.out.println("CH:"+t.getChannel());
        t.gotoPrevChannel();
        System.out.println("CH:"+t.getChannel());
        t.gotoPrevChannel();
        System.out.println("CH:"+t.getChannel());
    }
}
```

```
private int prevChannel; // 이전 채널

public void setChannel(int channel) {
    ....

    prevChannel = this.channel; // 현재 채널 저장
}

public void gotoPrevChannel() {
    setChannel(prevChannel);
}
```

[실행결과]

```
CH:10
CH:20
CH:10
CH:20
```

[7-12] 다음 중 접근 제어자에 대한 설명으로 옳지 않은 것은? (모두 고르시오)

- a. public은 접근제한이 전혀 없는 접근 제어자이다.
- b. (default)가 붙으면, 같은 패키지 내에서만 접근이 가능하다.
- ☒ c. 지역변수에도 접근 제어자를 사용할 수 있다.
- d. protected가 붙으면, 같은 패키지 내에서도 접근이 가능하다.
- e. protected가 붙으면, 다른 패키지의 자손 클래스에서 접근이 가능하다.

제어자	같은 클래스	같은 패키지	자손클래스	전 체
public				
protected				
default				
private				

[7-13] Math클래스의 생성자는 접근 제어자가 private이다. 그 이유는 무엇인가?

Math 클래스의 모든 메소드가 static 메소드이고, 인스턴스 변수가 존재하지 않기 때문에 객체를 생성할 필요가 없다.

[7-14] 문제7-1에 나오는 섯다카드의 숫자와 종류(isKwang)는 사실 한번 값이 지정되면 변경되어서는 안 되는 값이다. 카드의 숫자가 한번 잘못 바뀌면 똑같은 카드가 두 장이 될 수도 있기 때문이다. 이러한 문제점이 발생하지 않도록 아래의 SutdaCard를 수정하시오.

【연습문제】/ch7/Exercise7_14.java

```
class SutdaCard {
    int num;          final int NUM;
    boolean isKwang;  final boolean IS_KWANG;

    SutdaCard() {
        this(1, true);
    }

    SutdaCard(int num, boolean isKwang) {
        NUM      this.num = num;
        IS_KWANG this.isKwang = isKwang;
    }

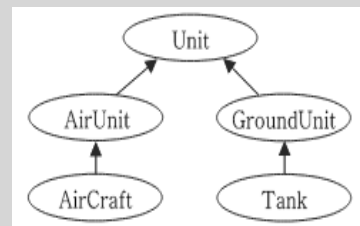
    public String toString() {
        return num + (isKwang ? "K":""");
    }
}

class Exercise7_14 {
    public static void main(String args[]) {
        SutdaCard card = new SutdaCard(1, true);
    }
}
```

[7-15] 클래스가 다음과 같이 정의되어 있을 때, 형변환을 올바르게 하지 않은 것은?
(모두 고르시오.)

```
class Unit {}
class AirUnit extends Unit {}
class GroundUnit extends Unit {}
class Tank extends GroundUnit {}
class AirCraft extends AirUnit {}

Unit u = new GroundUnit();
Tank t = new Tank();
AirCraft ac = new AirCraft();
```



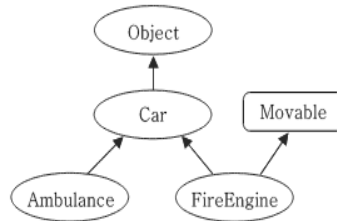
- a. u = (Unit)ac;
- b. u = ac;
- c. GroundUnit gu = (GroundUnit)u;
- d. AirUnit au = ac;
- e. t = (Tank)u; 부모 타입의 인스턴스를 자식 타입으로 형변환 할 수 없다.
- f. GroundUnit gu = t;

[7-16] 다음 중 연산결과가 true가 아닌 것은? (모두 고르시오)

```
class Car {}
class FireEngine extends Car implements Movable {}
class Ambulance extends Car {}

FireEngine fe = new FireEngine();
```

- a. fe instanceof FireEngine
- b. fe instanceof Movable
- c. fe instanceof Object
- d. fe instanceof Car
- ☒ e. fe instanceof Ambulance



instanceof 연산자는 실제 인스턴스의 모든 부모나 구현한 인터페이스에 대해 true를 반환한다.
 형변환이 가능한지 여부 확인 ex) parent instanceof Child : 부모가 자식집을 찾았으니 false가 된다.
 형변환이 불가능한 즉 타입이 상위 클래스도 하위 클래스도 아닐 경우에는 예러가 난다.

[7-17] 아래 세 개의 클래스로부터 공통부분을 뽑아서 Unit이라는 클래스를 만들고, 이 클래스를 상속받도록 코드를 변경하시오.

```
class Marine {    // 보병
    int x, y;    // 현재 위치
    void move(int x, int y) { /* 지정된 위치로 이동 */ }
    void stop()    { /* 현재 위치에 정지 */ }
    void stimPack() { /* 스팀팩을 사용한다. */ }
}

class Tank {      // 탱크
    int x, y;    // 현재 위치
    void move(int x, int y) { /* 지정된 위치로 이동 */ }
    void stop()    { /* 현재 위치에 정지 */ }
    void changeMode() { /* 공격모드를 변환한다. */ }
}

class Dropship { // 수송선
    int x, y;    // 현재 위치
    void move(int x, int y) { /* 지정된 위치로 이동 */ }
    void stop()    { /* 현재 위치에 정지 */ }
    void load()    { /* 선택된 대상을 태운다. */ }
    void unload()  { /* 선택된 대상을 내린다. */ }
}
```

```
abstract class Unit {
    int x, y;
    abstract void move(int x, int y);
    void stop() {}
}
```

형광칠 한 부분을 삭제하고, Marine, Tank, Dropship에 extends Unit을 추가한다.

[7-18] 다음과 같은 실행결과를 얻도록 코드를 완성하십시오.

[Hint] instanceof 연산자를 사용해서 형변환한다.

메서드명 : action

기능 : 주어진 객체의 메서드를 호출한다.

DanceRobot인 경우, dance()를 호출하고,

SingRobot인 경우, sing()을 호출하고,

DrawRobot인 경우, draw()를 호출한다.

반환타입 : 없음

매개변수 : Robot r - Robot인스턴스 또는 Robot의 자손 인스턴스

[연습문제]/ch7/Exercise7_18.java

```
class Exercise7_18 {
    /*
        (1) action메서드를 작성하십시오.
    */

    public static void main(String[] args) {
        Robot[] arr = { new DanceRobot(), new SingRobot(), new DrawRobot() };

        for(int i=0; i< arr.length;i++)
            action(arr[i]);
    } // main

    class Robot {}

    class DanceRobot extends Robot {
        void dance() {
            System.out.println("춤을 춥니다.");
        }
    }

    class SingRobot extends Robot {
        void sing() {
            System.out.println("노래를 합니다.");
        }
    }

    class DrawRobot extends Robot {
        void draw() {
            System.out.println("그림을 그립니다.");
        }
    }

    public static void action(Robot r) {
        if (r instanceof DanceRobot) {
            DanceRobot dr = (DanceRobot) r;
            dr.dance();
        } else if (r instanceof SingRobot) {
            SingRobot ar = (SingRobot) r;
            sr.sing();
        } else if (r instanceof DrawRobot) {
            DrawRobot dr = (DrawRobot) r;
            dr.draw();
        }
    }
}
```

[실행결과]

```
춤을 춥니다.
노래를 합니다.
그림을 그립니다.
```

[7-19] 다음은 물건을 구입하는 사람을 정의한 Buyer 클래스이다. 이 클래스는 멤버변수로 돈(money)과 장바구니(cart)를 가지고 있다. 제품을 구입하는 기능의 buy메서드와 장바구니에 구입한 물건을 추가하는 add메서드, 구입한 물건의 목록과 사용금액, 그리고 남은 금액을 출력하는 summary메서드를 완성하시오.

1. 메서드명 : buy

기능 : 지정된 물건을 구입한다. 가진 돈(money)에서 물건의 가격을 빼고,
장바구니(cart)에 담는다.
만일 가진 돈이 물건의 가격보다 적다면 바로 종료한다.

반환타입 : 없음

매개변수 : Product p - 구입할 물건

2. 메서드명 : add

기능 : 지정된 물건을 장바구니에 담는다.
만일 장바구니에 담을 공간이 없으면, 장바구니의 크기를 2배로 늘린 다음에 담는다.

반환타입 : 없음

매개변수 : Product p - 구입할 물건

3. 메서드명 : summary

기능 : 구입한 물건의 목록과 사용금액, 남은 금액을 출력한다.

반환타입 : 없음

매개변수 : 없음

[연습문제]/ch7/Exercise7_19.java

```
class Exercise7_19 {
    public static void main(String args[]) {
        Buyer b = new Buyer();
        b.buy(new Tv());
        b.buy(new Computer());
        b.buy(new Tv());
        b.buy(new Audio());
        b.buy(new Computer());
        b.buy(new Computer());
        b.buy(new Computer());

        b.summary();
    }
}

class Buyer {
    int money = 1000;
    Product[] cart = new Product[3]; // 구입한 제품을 저장하기 위한 배열
    int i = 0; // Product배열 cart에 사용될 index

    void buy(Product p) {
        /*
        (1) 아래의 로직에 맞게 코드를 작성하시오.
        1.1 가진 돈과 물건의 가격을 비교해서 가진 돈이 적으면 메서드를 종료한다.
        1.2 가진 돈이 충분하면, 제품의 가격을 가진 돈에서 빼고
        1.3 장바구니에 구입한 물건을 담는다. (add메서드 호출)

        if (money < p.price) {
            System.out.println("잔액 부족" + p + "을 살 수 없습니다.");
        } else {
            money -= p.price;
            add(p);
        }
        */
    }
}
```

```

    */
}

void add(Product p) {
    /*
    (2) 아래의 로직에 맞게 코드를 작성하시오.
    1.1 i의 값이 장바구니의 크기보다 같거나 크면
        1.1.1 기존의 장바구니보다 2배 큰 새로운 배열을 생성한다.
        1.1.2 기존의 장바구니의 내용을 새로운 배열에 복사한다.
        1.1.3 새로운 장바구니와 기존의 장바구니를 바꾼다.
    1.2 물건을 장바구니(cart)에 저장한다. 그리고 i의 값을 1 증가시킨다.
    */
} // add(Product p)

void summary() {
    /*
    (3) 아래의 로직에 맞게 코드를 작성하시오.
    1.1 장바구니에 담긴 물건들의 목록을 만들어 출력한다.
    1.2 장바구니에 담긴 물건들의 가격을 모두 더해서 출력한다.
    1.3 물건을 사고 남은 금액(money)를 출력한다.
    */
} // summary()

class Product {
    int price;           // 제품의 가격

    Product(int price) {
        this.price = price;
    }
}

class Tv extends Product {
    Tv() { super(100); }

    public String toString() { return "Tv"; }
}

class Computer extends Product {
    Computer() { super(200); }

    public String toString() { return "Computer"; }
}

class Audio extends Product {
    Audio() { super(50); }

    public String toString() { return "Audio"; }
}

```

```

    if (i >= cart.length) {
        Product[] tmp = new Product[cart.length*2];
        System.arraycopy(cart, 0, tmp, 0, cart.length);
        cart = tmp;
    }
    cart[i++] = p;

    String itemList = "";
    int sum = 0;

    for (int i=0 ; i<cart.length ; i++) {
        if (cart[i] == null) {
            break;
        }
        itemList += cart[i] + ", ";
        sum += cart[i].price;
    }

    System.out.println("구입 물건 : " + itemList);
    System.out.println("사용 금액 : " + sum);
    System.out.println("남은 금액 : " + money);

```

[실행결과]

```

잔액이 부족하여 Computer을/를 살수 없습니다.
구입한 물건: Tv, Computer, Tv, Audio, Computer, Computer,
사용한 금액: 850
남은 금액: 150

```

[7-20] 다음의 코드를 실행한 결과를 적으시오.

[연습문제]/ch7/Exercise7_20.java

```
class Exercise7_20 {
    public static void main(String[] args) {
        Parent p = new Child();
        Child c = new Child();

        System.out.println("p.x = " + p.x);
        p.method();

        System.out.println("c.x = " + c.x);
        c.method();
    }
}

class Parent {
    int x = 100;

    void method() {
        System.out.println("Parent Method");
    }
}

class Child extends Parent {
    int x = 200;

    void method() {
        System.out.println("Child Method");
    }
}
```

[실행결과]

p.x = 100
Child Method
c.x = 200
Child Method

부모 클래스에 선언된 멤버변수와 같은 이름의 인스턴스 변수를 자식 클래스에 중복으로 정의했을 때, 부모 타입의 참조 변수로 자식 인스턴스를 참조하는 경우와 자식 타입의 참조 변수로 자식 인스턴스를 참조하는 경우는 서로 다른 결과를 얻는다.

[7-21] 다음과 같이 attack메서드가 정의되어 있을 때, 이 메서드의 매개변수로 가능한 것 두 가지를 적으시오.

```
interface Movable {
    void move(int x, int y);
}

void attack(Movable f) {
    /* 내용 생략 */
}
```

null, Movable 인터페이스를 구현한 클래스 또는 그 자손의 인스턴스

[7-22] 아래는 도형을 정의한 Shape클래스이다. 이 클래스를 조상으로 하는 Circle클래스와 Rectangle클래스를 작성하시오. 이 때, 생성자도 각 클래스에 맞게 적절히 추가해야 한다.

- (1) 클래스명 : Circle
 조상클래스 : Shape
 멤버변수 : double r - 반지름
- (2) 클래스명 : Rectangle
 조상클래스 : Shape
 멤버변수 : double width - 폭
 double height - 높이
- 메서드 :
- 메서드명 : isSquare
 기능 : 정사각형인지 아닌지를 알려준다.
 반환타입 : boolean
 매개변수 : 없음

[연습문제]/ch7/Exercise7_22.java

```
abstract class Shape {
    Point p;

    Shape() {
        this(new Point(0,0));
    }

    Shape(Point p) {
        this.p = p;
    }

    abstract double calcArea(); // 도형의 면적을 계산해서 반환하는 메서드

    Point getPosition() {
        return p;
    }

    void setPosition(Point p) {
        this.p = p;
    }
}
```

```
class Point {
    int x;
    int y;

    Point() {
        this(0,0);
    }

    Point(int x, int y) {
        this.x=x;
        this.y=y;
    }
}
```

class Rect extends Shape {

```
    double width;
    double height;
```

```
    Rect(double width, double height) {
        this(new Point(0, 0), width, height);
    }
```

```
    Rect(Point p, double width, double height) {
        super(p); // 부모의 멤버는 부모의 생성자가 초기화 하도록 한다
        this.width = width;
        this.height = height;
    }
```

```
    boolean isSquare() {
        return width*height !=0 && width == height;
    }
```

```
    double calcArea() {
        return width*height;
    }
```

}

class Circle extends Shape {
 double r;

```
    Circle(double r) {
        this(new Point(0, 0), r);
    }
```

```
    Circle(Point p, double r) {
        super(p);
        this.r = r;
    }
```

```
    double calcArea() {
        return Math.PI * r * r;
    }
```

}

```

    }

    public String toString() {
        return "["+x+","+y+"]";
    }
}

```

[7-23] 문제7-22에서 정의한 클래스들의 면적을 구하는 메서드를 작성하고 테스트 하시오.

1. 메서드명 : sumArea

기능 : 주어진 배열에 담긴 도형들의 넓이를 모두 더해서 반환한다.

반환타입 : double

매개변수 : Shape[] arr

[연습문제]/ch7/Exercise7_23.java

```

class Exercise7_23
{
    /*
        (1) sumArea메서드를 작성하시오.
    */

    public static void main(String[] args)
    {
        Shape[] arr = {new Circle(5.0), new Rectangle(3,4), new Circle(1)};
        System.out.println("면적의 합:"+sumArea(arr));
    }
}

static double sumArea(Shape[] arr) {
    double sum = 0;

    for (int i=0; i<arr.length; i++) {
        sum += arr[i].calcArea();
    }

    return sum;
}

```

[실행결과]

면적의 합:93.68140899333463

[7-24] 다음 중 인터페이스의 장점이 아닌 것은?

- a. 표준화를 가능하게 해준다.
- b. 서로 관계없는 클래스들에게 관계를 맺어 줄 수 있다.
- c. 독립적인 프로그래밍이 가능하다.
- d. 다중상속을 가능하게 해준다.
- ❖ e. 패키지간의 연결을 도와준다.

1. 개발시간을 단축시킬 수 있다.

알단 인터페이스가 작성되면, 이를 사용해서 프로그램을 작성하는 것이 가능하다. 메서드를 호출하는 쪽에서는 메서드의 내용에 관계없이 선언부만 알면 되기 때문이다.

그리고 동시에 다른 한 쪽에서는 인터페이스를 구현하는 클래스를 작성하도록 하여, 인터페이스를 구현하는 클래스가 작성될 때까지 기다리지 않고도 양쪽에서 동시에 개발을 진행할 수 있다.

2. 표준화가 가능하다.

프로젝트에 사용되는 기본 틀을 인터페이스로 작성한 다음, 개발자들에게 인터페이스를 구현하여 프로그램을 작성하도록 함으로써 보다 일관되고 정형화된 프로그램의 개발이 가능하다.

3. 서로 관계없는 클래스들에게 관계를 맺어 줄 수 있다.

서로 상속관계에 있지도 않고, 같은 조상클래스를 가지고 있지 않은 서로 아무런 관계도 없는 클래스들에게 하나의 인터페이스를 공통적으로 구현하도록 함으로써 관계를 맺어 줄 수 있다.

4. 독립적인 프로그래밍이 가능하다.

인터페이스를 이용하면 클래스의 선언과 구현을 분리시킬 수 있기 때문에 실제구현에 독립적인 프로그램을 작성하는 것이 가능하다. 클래스와 클래스간의 직접적인 관계를 인터페이스를 이용해서 간접적인 관계로 변경하면, 한 클래스의 변경이 관련된 다른 클래스에 영향을 미치지 않는 독립적인 프로그래밍이 가능하다.

[7-25] Outer 클래스의 내부 클래스 Inner의 멤버변수 iv의 값을 출력하시오.

[연습문제]/ch7/Exercise7_25.java

```
class Outer {
    class Inner {
        int iv=100;
    }
}

class Exercise7_25 {
    public static void main(String[] args) {
        /*
            (1) 알맞은 코드를 넣어 완성하시오.
        */
    }
}
```

Outer o = new Outer();
Outer.Inner ii = o.new Inner();
System.out.println(ii.iv);

[실행결과]

100

내부 클래스의 인스턴스를 생성하기 위해서는 외부 클래스의 인스턴스를 먼저 생성해야 한다.

[7-26] Outer 클래스의 내부 클래스 Inner의 멤버변수 iv의 값을 출력하시오.

[연습문제]/ch10/Exercise7_26.java

```
class Outer {
    static class Inner {
        int iv=200;
    }
}

class Exercise7_26 {
    public static void main(String[] args) {
        /*
            (1) 알맞은 코드를 넣어 완성하시오.
        */
    }
}
```

Outer.Inner ii = new Outer.Inner();
System.out.println(ii.iv);

[실행결과]

200

static 클래스는 인스턴스 클래스와 달리 외부 클래스의 인스턴스를 생성하지 않고도 사용할 수 있다.

[7-27] 다음과 같은 실행결과를 얻도록 (1)~(4)의 코드를 완성하시오.

[연습문제]/ch7/Exercise7_27.java

```
class Outer {
    int value=10;

    class Inner {
        int value=20;
        void method1() {
            int value=30;

            System.out.println(/* (1) */);
            System.out.println(/* (2) */);
            System.out.println(/* (3) */);
        }
    } // Inner클래스의 끝
} // Outer클래스의 끝

class Exercise7_27 {
    public static void main(String args[]) {
        /*
            (4) 알맞은 코드를 넣어 완성하시오.
        */

        inner.method1();
    }
}
```

value
this.value
Outer.this.value

Outer outer = new Outer();
Outer.Inner inner = outer.new Inner();

[실행결과]

```
30
20
10
```

[7-28] 아래의 EventHandler를 익명 클래스(anonymous class)로 변경하시오.

[연습문제]/ch7/Exercise7_28.java

```
import java.awt.*;
import java.awt.event.*;

class Exercise7_28
{
    public static void main(String[] args)
    {
        Frame f = new Frame();
        f.addWindowListener(new EventHandler());
    }
}

class EventHandler extends WindowAdapter
{
    public void windowClosing(WindowEvent e) {
        e.getWindow().setVisible(false);
        e.getWindow().dispose();
        System.exit(0);
    }
}
```

??

```
f.addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent e) {
        e.getWindow().setVisible(false);
        e.getWindow().dispose();
        System.exit(0);
    }
});
```

[7-29] 지역 클래스에서 외부 클래스의 인스턴스 멤버와 static멤버에 모두 접근할 수 있지만, 지역변수는 final이 붙은 상수만 접근할 수 있는 이유 무엇인가?

메소드가 수행을 마쳐 지역 변수가 소멸된 시점에도, 지역 클래스의 인스턴스가 소멸된 지역 변수를 참조하려는 경우가 발생할 수 있기 때문이다.