제 19 강 : 추상클래스

※ 학습목표

✓ 추상 메서드의 특징을 설명할 수 있다.
✓ 추상 클래스를 작성할 수 있다.
✓ 추상 클래스의 상속관계를 설명할 수 있다.
✓ 추상클래스와 다형성을 활용할 수 있다.

1. 추상클래스

① 추상화의 이해와 선언법

✓ 추상화라는 것은 구체적인 개념으로부터 공통된 부분들만 추려내어 일반화 할 수 있도록 하는 것을 의미한다.
✓ 다시 말해서 일반적으로 사용할 수 있는 단계가 아닌 아직 미완성(未完成)적 개념인 것이다.
✓ 그럼 자바에서 얘기하는 추상(abstract)화 작업을 하기 위해서는 먼저 추상 메소드를 이해해야 한다.

✓ 추상 메소드의 구성

```
public abstract void abstractMethod( );
```

✓ 메소드를 정의하면서 brace({})를 생략하여 실상 메소드가 하는 일(body)이 없이 semicolon(;)으로 문장의 끝을 나타내었다.
✓ 그리고 abstract라는 예약어를 통해 현 메소드가 추상 메소드임을 명시하였다.
✓ 추상 메소드를 하나라도 가지게 되는 클래스가 바로 추상 클래스가 된다.
✓ 그리고 이런 추상 클래스 또한 다음과 같이 추상 클래스임을 명시해야 한다.

```
public abstract class AbstractClass{ }
```

✓ 추상클래스는 객체를 만들 수 없다.

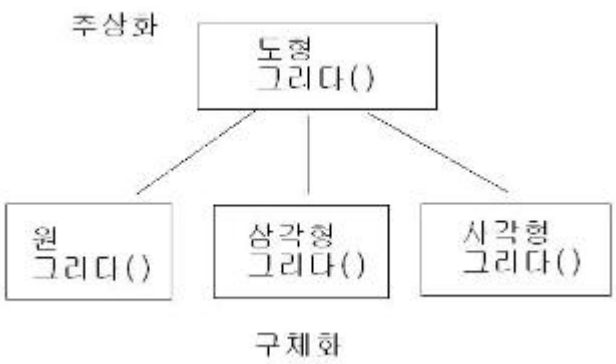
2. 추상화 작업

✓ 상속이 자손클래스를 만드는데 조상클래스를 사용하는 것이라면, 추상화는 기존의 클래스의 공통부분을 뽑아 내서 조상클래스를 만드는 것이라고 할 수 있다.

✓ 상속계층도를 따라 내려 갈수록 세분화되며, 올라갈수록 공통요소만 남게 된다.

✓ 추상화 - 클래스간의 공통점을 찾아내서 공통의 조상을 만드는 작업

✓ 구체화 - 상속을 통해 클래스를 구현, 확장하는 작업



[실습]

```
1  package tommy.java.exam01;
2
3  abstract class Diagram {
4          abstract void draw();
5  }
6  /*
7          private를 선언 할 수 없는 이유 :
8                  자식이 오버라이드 해야 하는데 못하게 된다.
9          static으로 선언 할 수 없는 이유 :
10                 객체 없이도 호출이 되는 메소드 이므로 반드시 body가
11                 정의 되어야 클래스.메서드() 형식으로 호출 할 수 있다.
12 */
13 class Triangle extends Diagram {
14         void draw() {
15                 System.out.println("삼각형을 그린다.");
16         }
17 }
18
19 class Rectangle extends Diagram {
20         void draw() {
21                 System.out.println("사각형을 그린다.");
22         }
23 }
24
```

```
25  class Circle extends Diagram {
26          void draw() {
27                  System.out.println("원을 그린다.");
28          }
29  }
30
31  public class UseDraw {
32          public static void main(String[] args) {
33                  Diagram[] ref = new Diagram[3];
34                  ref[0] = new Triangle();
35                  ref[1] = new Circle();
36                  ref[2] = new Rectangle();
37                  for (int i = 0; i < ref.length; i++)
38                          ref[i].draw();
39          }
40  }
```

3. 추상클래스의 상속관계

✓ 추상 클래스들 간에도 상속이 가능하다.

✓ 일반 클래스들 간의 상속과 유사하지만 추상 클래스들 간의 상속에서는 상속 받은 추상 메소드를 꼭 Override(재정의)할 필요는 없다.

✓ 그냥 상속만 받아두고 있다가 언제가 일반 클래스와 상속관계가 이루어 질 때가 있을 것이다. 이때 재정의 하지 못했던 상속 받은 추상 메소드를 모두 일반 클래스 내에서 재정의해도 되기 때문이다

[실습]

```
1   package tommy.java.exam02;
2
3   abstract class AbsEx1 {
4           int a = 100; // 변수
5           final String str = "abstract test"; // 상수
6
7           public String getStr() { // 일반 메소드
8                   return str;
9           }
10
11          // 추상 메소드는 몸체(body)가 없다.
12          abstract public int getA();
13  }
14
15  abstract class AbsEx2 extends AbsEx1 {
16          public int getA() { // 부모클래스의 추상 메소드 재 정의
```

```
17                      return a;
18              }
19
20              public abstract String getStr();
21      }
22
23      public class AbsEx extends AbsEx2 {
24              public String getStr() { // AbsEx2의 추상 메소드 재 정의
25                      return str; // str은 AbsEx1의 멤버이다
26              }
27
28              public static void main(String[] args) {
29                      AbsEx ae = new AbsEx();
30                      System.out.println("ae.getA():" + ae.getA());
31                      System.out.println("ae.getStr():" + ae.getStr());
32              }
33      }
```

4. 추상클래스 종합예제

✓ 같은 패키지 안에서 작업을 수행할 것.

✓ 아래 클래스들의 공통점을 찾아서 유닛을 이동시키는 프로그램을 만들어 보자.

Marine : 보병	SiegeTank : 시즈탱크	Dropship : 드랍쉽
현재위치:int x, y; **위치이동:** **void move** **(int x, int y){}** **위치정지:** **void stop(){}** **메시지 :** **void message(){}** 스팀팩을사용: void stimPack() {}	**현재위치:int x, y;** **위치이동:** **void move** **(int x, int y){}** **위치정지:** **void stop(){}** **메시지 :** **void message(){}** 시즈모드변환 : void changeMode() {}	**현재위치:int x, y;** **위치이동:** **void move** **(int x, int y){}** **위치정지:** **void stop(){}** **메시지 :** **void message(){}** 탑승시킨다. void load(){} 드랍시킨다. void dropdown(){}

✓ 공통요소를 모아서 Unit이라는 추상클래스를 작성하자.

```
1  package tommy.java.exam03;
2
3  public abstract class Unit {
4          int x, y;
5
6          abstract void move(int x, int y);
7
8          abstract void stop();
9
10         abstract void message();
11 }
```

✓ Unit을 상속받은 Marin 클래스 작성

```
1  package tommy.java.exam03;
2
3  public class Marine extends Unit {
4          void move(int x, int y) {
5                  System.out.println("마린의 위치 이동 좌표는 x : " +
                                   x + ", y : " + y + " 입니다.");
6          }
7
8          void stop() {
9                  System.out.println("마린이 대기 상태에 있습니다.");
10         }
11
12         void message() {
13                 System.out.println("Message:: Standing back.");
14         }
15
16         void stimPack() {
17                 System.out.println("마린이 스팀팩을 사용한다.");
18         }
19 }
```

✓ Unit을 상속받은 SiegeTank 클래스 작성

```
1  package tommy.java.exam03;
2
3  public class SiegeTank extends Unit {
4          void move(int x, int y) {
5                  System.out.println("Move it! Move it! SiageTank => x : " + x + ", y : " + y);
6          }
7
```

```
8           void stop() {
9                   System.out.println("Destination?");
10          }
11
12          void message() {
13                  System.out.println("Message::Go, Siege!");
14          }
15
16          void changeMode() {
17                  System.out.println("Yes, Sir!");
18          }
19  }
```

✓ Unit을 상속받은 Dropship 클래스 작성

```
1   package tommy.java.exam03;
2
3   public class Dropship extends Unit {
4           void move(int x, int y) {
5                   System.out.println("Take it slow. Droshp => x : " + x + ", y : " + y);
6           }
7
8           void stop() {
9                   System.out.println("In the by, by, by, by");
10          }
11
12          void message() {
13                  System.out.println("Message:: Can I take orders.");
14          }
15  }
```

✓ 다형성을 이용하여 유닛을 이동시키는 메인 클래스 작성

```
1   package tommy.java.exam03;
2
3   public class UseGame {
4           public static void main(String[] args) {
5                   Unit[] ref = new Unit[3];
6                   ref[0] = new Marine();
7                   ref[1] = new SiegeTank();
8                   ref[2] = new Dropship();
9                   for (int i = 0; i < ref.length; i++) {
10                          System.out.println("====================");
11                          // 모든 유닛을 이동 시키기.
12                          ref[i].move(100, 200);
```

```
13                                  ref[i].message();
14                                  System.out.println("====================");
15                          }
16                  }
17  }
```