제 17 강 : 상속

※ 학습목표
✓ 상속의 개념을 설명할 수 있다.
✓ 오버라이딩을 작성할 수 있다.
✓ 오버로딩과 오버라이딩의 차이를 설명할 수 있다.
✓ super, super() 포인터를 활용할 수 있다.

1. 클래스 상속 개념

용어	설명
기본 클래스 : Base Class 슈퍼 클래스 : Super Class 부모 클래스 : Parent Class	왼쪽의 용어가 모두 같은 뜻을 나타내며 상속을 주기위해 준비된 특정 클래스를 의미한다.
유도 클래스 : Derivation Class 하위 클래스 : Sub Class 자식 클래스 : Child Class	왼쪽의 용어가 모두 같은 뜻을 나타내며 특정 클래스로부터 상속을 받아 새롭게 정의 되는 클래스를 의미한다.

① 상속의 개념과 중요성
✓ 부모가 보유하고 있는 재산 중 일부를 자식이 물려받는 것을 의미한다.
✓ 자바에서는 이런 클래스들 간의 다중 상속을 지원하지 않으므로 객체의 명확성을 높였다

② 클래스 상속의 정의 법
✓ 자바에서 얘기하는 상속이라는 것은 특정 클래스가 가지는 일부 속성과 기능을 다른 새로운 클래스에게 제공하기 위해 맺는 클래스들 간의 관계를 말한다.
✓ 이는 super클래스를 새로운 sub클래스에서 [extends]라는 예약어를 사용하여 서로 관계를 맺은 상태이다

③ 클래스 상속의 중요성
✓ 클래스 상속은 객체의 재사용이라는 장점뿐만 아니라 코드의 간결성을 제공해 주는 객체 지향적 언어의 장점과 중요한 특징이 된다.

✓ 그러므로 잘 정의된 super클래스가 있다면 sub클래스의 작성이 간편해지고 무엇보다 개발 시간이 단축된다는데 상속의 중요성과 장점을 들 수 있다.

[실습]

```
1  package tommy.java.exam01;
2
3  class CellPhone {
4          String model;
5          String number;
6          int chord;
7          public void setNumber(String n) {
8                  number = n;
9          }
10         public String getModel() {
11                 return model;
12         }
13         public int getChord() {
14                 return chord;
15         }
16         public String getNumber() {
17                 return number;
18         }
19 }
20
21 public class MP3Phone extends CellPhone {
22         int size; // 저장 용량
23
24         public MP3Phone(String model, String num, int chord, int size) {
25                 this.model = model;
26                 number = num;
27                 this.chord = chord;
28                 this.size = size;
29         }
30 }
```

2. 오버라이딩

✓ 오버라이딩은 [메서드 재정의]라고도 불리며 이는 서로 상속관계로 이루어진 객체들 간의 관계에서 비롯된다.

✓ super클래스가 가지는 메서드를 sub클래스에서 똑 같은 것을 새롭게 만들게 되면 더 이상 super클래스의 이름이 같은 메서드를 호출할 수 없게 된다. 이를 Overriding 이라 하고 또는 멤버 은폐라고도 한다.

✓ 자손클래스에서 오버라이딩하는 메서드는 조상클래스의 메서드와
- ✓ 이름이 같아야 함
- ✓ 매개변수가 같아야 함
- ✓ 리턴타입이 같아야 함
- ✓ 접근 제한자는 더 넓게 지정해야 함

✓ 오버로딩과 오버라이딩의 차이점

차이점	오버라이딩	오버로딩
영역	상속간의 클래스들(최소 2개 이상)	하나의 클래스
메서드명	똑같아야 함	똑같아야 함
인자	똑같아야 함	반드시 달라야 함
리턴값	똑같아야 함	달라도 됨 (같아도 상관없음)
접근제한자	달라도 되나, 자식 클래스가 부모보다 넓게	달라도 됨 (같아도 상관없음)

[실습]

```
1  package tommy.java.exam02;
2
3  class CellPhone {
4          public void call() {
5                  System.out.println("통화를 합니다.");
6          }
7  }
8
9  public class CellPhone3G extends CellPhone {
10         public void call() {
11                 System.out.println("영상통화를 합니다.");
12         }
13 }
```

3. Object Class

✓ 한국 사람으로 보면 '단군'과 같은 존재

✓ 모든 클래스의 super클래스(배열을 제외하고 상속 받음)

✓ 'extends' 라는 키워드를 쓰지 않음

✓ ex) import java.lang.*과 같이 내부적으로 선언되어 있음

✓ 주요 메서드

반환형	메서드명	설 명
boolean	equals (Object obj)	obj와 같은지 검사하여 같으면 true (String,Wrapper 제외시 주소값 비교(==)와 같음)
Class <? extends Object>	getClass()	현 객체를 실행할 때 클래스를 반환함
int	hashCode()	현 객체의 해시코드를 반환함
String	toString()	현 객체의 파생 클래스명과 @에 이어 해쉬코드를 출력

[실습] 객체 비교 예제

```
1  package tommy.java.exam03;
2
3  public class ObjectEx {
4          private String name;
5          private int price;
6          public ObjectEx(String name, int price) {
7                  this.name = name;
8                  this.price = price;
9          }
10         public static void main(String[] args) {
11                 ObjectEx test1 = new ObjectEx("1", 1);
12                 ObjectEx test2 = new ObjectEx("1", 1);
13                 System.out.println("test1은? " + test1);
14                 System.out.println("test2은? " + test2);
15                 System.out.println("test1의 해시코드는? " + test1.hashCode());
16                 System.out.println("test2의 해시코드는? " + test2.hashCode());
17                 System.out.println("test1과 test2는 같은가? " + test1.equals(test2));
18         }
19 }
```

[실습] 객체 비교 예제 : 해시코드와 주소 값의 차이

```
1  package tommy.java.exam04;
2
3  public class HashcodeEx {
4          public static void main(String[] args) {
5                  String str = new String("TEST");
6                  String str2 = new String("TEST");
7                  System.out.println("str과 str2의 주소값은 같나요 ? : " + (str == str2));
8                  System.out.println("str과 str2의 해시코드는 같나요? : " +
                                          (str.hashCode() == str.hashCode()));
9                  System.out.println("str의 해시코드? : " + str.hashCode());
10                 System.out.println("str2의 해시코드? : " + str2.hashCode());
11                 // 객체의 해시코드는?
12                 HashcodeEx test1 = new HashcodeEx();
13                 HashcodeEx test2 = new HashcodeEx();
14                 System.out.println("test1과 test2의 주소값은 같나요 ? : " + (test1 == test2));
15                 System.out.println("test1과 test2의 해시코드값은 같나요? : " +
                                          (test1.hashCode() == test2.hashCode()));
16                 System.out.println("test1의 해시코드는 ? : " + test1.hashCode());
17                 System.out.println("test2의 해시코드는 ? : " + test2.hashCode());
18         }
19 }
```

4. super 와 super()

① super

✓ 부모 객체의 레퍼런스

✓ 부모클래스의 멤버와 자손클래스의 멤버가 중복 정의되어 서로 구별해야 하는 경우에만 super를 사용하는 것이 좋음

[실습]

```
1  package tommy.java.exam05;
2
3  class Super {
4          int a = 5;
5  }
6
7  class Sub extends Super {
8          int a = 10;
9
10         public void test() {
11                 System.out.println(this.a);
12                 System.out.println(super.a);
```

```
13          }
14  }
15
16  public class SuperEx {
17          public static void main(String[] args) {
18                  Sub sub = new Sub();
19                  sub.test();
20          }
21  }
```

② super()

✓ 자식의 기초 생성자에는 super()가 생략되어 있음

✓ 인스턴스 생성시 무조건 부모 클래스부터 생성됨

✓ 즉, 부모의 생성자를 무조건 먼저 호출함

✓ this(), super()는 생성자의 가장 위에 명시해야 함

[실습]

```
1   package tommy.java.exam06;
2
3   class Super {
4           public Super(int x) {
5                   System.out.println("상위클래스 생성자 : " + x);
6           }
7   }
8
9   class Sub extends Super {
10          public Sub() {
11                  super(5);
12                  System.out.println("하위클래스 생성자");
13          }
14  }
15
16  public class SuperEx2 {
17          public static void main(String[] args) {
18                  Sub sub = new Sub();
19          }
20  }
```

③ 종합실습 : this(), super()

✓ 아래의 파일을 같은 패키지에 작성하고 실행한다.

✓ PointEx.java 작성

```
1  package tommy.java.exam07;
2
3  public class PointEx {
4          private int x;
5          private int y;
6
7          public PointEx() {
8          }
9
10         public PointEx(int x) {
11                 this.x = x;
12         }
13
14         public PointEx(int x, int y) {
15                 this(x);
16                 this.y = y;
17         }
18
19         public void setX(int x) {
20                 this.x = x;
21         }
22
23         public void setY(int y) {
24                 this.y = y;
25         }
26
27         public int getX() {
28                 return x;
29         }
30
31         public int getY() {
32                 return y;
33         }
34
35         public void disp() {
36                 System.out.println("x value is " + x);
37                 System.out.println("y value is " + y);
38         }
39 }
```

✓ CircleEx.java 작성

```
1  package tommy.java.exam07;
2
3  public class CircleEx extends PointEx {
4          private int r;
5
6          public CircleEx() {
7          }
8
9          public CircleEx(int x) {
10                 super(x);
11                 r = 1;
12         }
13
14         public CircleEx(int x, int y) {
15                 super(x, y);
16                 r = 2;
17         }
18
19         public CircleEx(int x, int y, int r) {
20                 super(x, y);
21                 this.r = r;
22         }
23
24         public void setR(int r) {
25                 this.r = r;
26         }
27
28         public int getR() {
29                 return r;
30         }
31
32         public void disp() {
33                 super.disp();
34                 System.out.println("r value is " + r);
35         }
36
37         public static void main(String[] ar) {
38                 CircleEx ce = new CircleEx();// 생성자를 바꿔보자.
39                 ce.disp();
40         }
41 }
```

5. 돌발퀴즈 : 상속개념 종합예제

✓ 앞의 16강에 예제의 Account 클래스를 상속받아 아래와 같은 NewAccount 클래스를 만들고 Banking.java의 main에 비밀번호 변경기능을 추가해서 새로운 은행 시스템을 만들어 보자

[힌트]

```
1  package tommy.java.exam08;
2
3  public class NewAccount extends Account {
4          // 멤버
5          private String pass;
6
7          // 생성자
8          public NewAccount() {
9          }
10
11         public NewAccount(String name, String pass) {
12                 super(name);
13                 this.pass = pass;
14         }
15
16         // 메소드
17         public boolean passCheck(String pass) {
18                 if (pass.equals(this.pass)) {
19                         return true;
20                 } else {
21                         return false;
22                 }
23         }
24
25         public void setPass(String pass) {
26                 this.pass = pass;
27         }
28
29         public String getPass() {
30                 return pass;
31         }
32 }
```