제3장 : 인스턴스 기본 조작

Object 클래스의 기본 기능

1. 모든 클래스는 Object 클래스의 메서드를 가지고 있다
2. Object 타입 변수에는 모든 인스턴스를 대입할 수 있다

<Object 클래스의 대표 메서드>

- toString() : 문자열 표현을 얻음
- equals() : 비교
- hashCode() : 해시값을 얻음

오버라이드하여 원하는 결과를 얻도록 수정할 수 있음

```
class Hero {
  private String name;
  private int hp;

  // getter, setter 생략

  @Override
  public String toString() {
    return "Hero(name : " + name + ", hp : " + hp;
  }
}
```

equals() 와 == 의 차이

- equals() : equals() 메서드 내에서 정의한 규칙에 의한 비교
- == : 참조의 비교

equals() 메서드도 오버라이드 하여 나만의 규칙을 정의할 수 있음

```
@Override
public boolean equals(Object o) {
  if (o == this) return true;
  if (o == null) return false;
  if (!(o instanceof Hero)) return false;
  Hero hero = (Hero) o;
  if (!(this.name.equals(hero.name))) {
    return false;
  }
  return true;
}
```

equals() 를 오버라이드 하지 않은 객체의 취급

Quiz

```
class Hero {
  public String name;
}


class Main {
  public static void main(String[] args) {
    List<Hero> list = new ArrayList<>();

    Hero hero = new Hero();
    hero.name = "홍길동";
    list.add(hero);
    System.out.println(list.size());   // ?

    hero = new Hero();
    hero.name = "홍길동";
    list.remove(hero);
    System.out.println(list.size());   // ?

  }
}
```

HashSet 에서 remove()

Quiz

```
class Hero {
  public String name;
  @Override
  public boolean equals(Object o) { ... };
}


class Main {
  public static void main(String[] args) {
    Set<Hero> heroSet = new HashSet<>();

    Hero hero = new Hero();
    hero.name = "홍길동";
    list.add(hero);
    System.out.println(list.size());   // ?

    hero = new Hero();
    hero.name = "홍길동";
    list.remove(hero);
    System.out.println(list.size());   // ?

  }
}
```

HashSet 의 동작 원리

Hash 계열은 요소를 검색할 때 equals() 보다 비용이 싼 hashCode() 비교를 사용함

1. 모든 객체는 해시값을 가진다
2. 동일한 객체는 항상 같은 해시값을 가진다.

hashCode() 의 오버라이드

Hash 계열은 요소를 검색할 때 equals() 보다 비용이 싼 hashCode() 비교를 사용함

1. 모든 객체는 해시값을 가진다
2. 동일한 객체는 항상 같은 해시값을 가진다.

```
@Override
public int hashCode() {
  int result = 37;  // 적당한 초기값
  result = result * 31 + name.hashCode();
  result = result * 31 + hp;
  return result;
}
```

인스턴스의 순서 결정

Collections.sort() 메서드는 컬렉션 내부를 정렬해 줌

```
List<Account> list = new ArrayList<>();

// ... 생략

Collections.sort(list);
```

하지만 sort() 메서드를 사용하기 위해서는 다음과 같은 제약이 따름

```
public static <T extends Comparable<? super T>> void sort(List<T> list)
```

Comparable 인터페이스 구현

제약) 정렬 대상은 반드시 Comparable 인터페이스를 구현해야 함

```
public class Account implements Comparable<Account> {
  int number;

  @Override
  public int compareTo(Account obj) {
    if (this.number < obj.number) {
      return -1;  // this가 더 작은 경우
    }
    if (this.number > obj.number) {
      return 1;   // this가 더 큰 경우
    }
    return 0;   // 같은 경우
  }
}
```

Quiz

```
Hero h1 = new Hero("홍길동");
Hero h2 = h1;
```

clone() 메서드


```
public class Hero implements Cloneable {
  public String name;
  public int hp;
  public Sword sword;

  @Override
  public Hero clone() {
    Here result = new Hero();
    result.name = this.name;
    result.hp = this.hp;
    result.sword = this.sword;
    return result;
  }
}
```

Quiz

```
Hero h1 = new Hero("홍길동");
Hero h2 = h1.clone();
```

```
public class Hero implements Cloneable {
  public String name;
  public int hp;
  public Sword sword;

  @Override
  public Hero clone() {
    Here result = new Hero();
    result.name = this.name;
    result.hp = this.hp;
    result.sword = this.sword.clone();
    return result;
  }
}
```

연습문제 3-1

다음과 같은 Book 클래스가 있습니다.

```
1  public class Book {
2    private String title;
3    private Date publishDate;
4    private String comment;
5
6    // getter/setter
7  }
```

다음 동작을 할 수 있도록 Book 클래스를 수정하시오.

1) 제목과 출간일이 같으면 같은 책으로 판단한다. 또한 HashSet 등의 컬렉션에 넣어도 동일 객체로 판단한다.
2) Book 인스턴스를 담고 있는 컬렉션에 대해 Collections.sort() 를 사용하면 출간일이 오래된 순서대로 정렬된다.
3) clone() 메서드를 호출하면 복제된다.