접근 제어

접근 지정자	클래스	패키지	자식 클래스	전체 세계
public	O	O	O	O
protected	O	O	O	X
없음	O	O	X	X
private	O	X	X	X

EmployeeTest.java

```
01  import java.util.*;
02  class Employee {
03      private String name;      // private로 선언
04      private int salary;       // private로 선언
05      int age;                  // package로 선언
06
07      // 생성자
08      public Employee(String n, int a, double s) {
09          name = n;
10          age = a;
11          salary = s;
12      }
13      // 직원의 이름을 반환
14      public String getName() {
15          return name;
16      }
17      // 직원의 월급을 반환
```

전용멤버

패키지 멤버

```
18      private int getSalary() {       // private로 선언
19          return salary;
20      }
21      // 직원의 나이를 반환
22      int getAge() {                  // package로 선언
23          return age;
24      }
25  }
26
27  public class EmployeeTest {
28      public static void main(String[] args) {
29          Employee e;
30          e = new Employee("홍길동", 0, 3000);
31          e.salary = 300;             // 오류! private 변수
32          e.age = 26;                 // 같은 패키지이므로 OK
33          int sa = e.getSalary();     // 오류! private 메소드
34          String s = e.getName();     // OK!
35          int a = e.getAge();         // 같은 패키지이므로 OK
36      }
37  }
```

설정자 & 접근자

설정자	접근자
setter	getter
필드의 값을 설정하는 메소드	필드의 값을 반환하는 메소드
소속변수에 대한 대입문 대신 사용	소속변수에 저장된 값을 읽어오는데 사용
public void set~~(....)	public 리턴타입 get~~()
ex) private int speed; public void setSpeed(int s)	ex) private int speed; public int getSpeed()

```
01  class Car {
02      private String color;          // 색상
03      private int speed;             // 속도
04      private int gear;              // 기어
05      public String getColor() {
06          return color;
07      }
08      public void setColor(String c) {
09          color = c;
10      }
11      public int getSpeed()          {       return speed;   }
12      public void setSpeed(int s)    {       speed = s;      }
13      public int getGear()           {       return gear;    }
14      public void setGear(int g)     {       gear = g;   }
15  }
```

필드가 모두 private로 선언되었다.
클래스 내부에서만 사용이 가능하다.

color에 대한 접근자 메소드

color에 대한 설정자 메소드

```
public static void main(String[] args)
{
    Car myCar = new Car();

    myCar.color = "red";
    myCar.speed = 100;
    myCar.gear = 1;

    System.out.print(myCar.color);
    System.out.print(myCar.speed);
    System.out.print(myCar.gear);
}
```

```
public static void main(String[] args)
{
  Car myCar = new Car();

  myCar.setColor("red");
  myCar.setSpeed(100);
  myCar.setGear(1);

  System.out.print(myCar.getColor());
  System.out.print(myCar.getSpeed());
  System.out.print(myCar.getGear());
}
```

- 설정자와 접근자는 왜 사용하는가?
  - 설정자에서 매개 변수를 통하여 잘못된 값이 넘어오는 경우, 이를 사전에 차단할 수 있다.
  - 필요할 때마다 필드값을 계산하여 반환할 수 있다.
  - 접근자만을 제공하면 자동적으로 읽기만 가능한 필드를 만들 수 있다.

```
public void setSpeed(int s)
{
    if( s < 0 )
        speed = 0;
    else
        speed = s;
}
```

속도가 음수이면 0으로 만든다.

접근 제어

```
public class Account {
        private int regNumber;
        private String name;
        private int balance;

        public String getName() {                   return name;          }
        public void setName(String name) {                  this.name = name;     }
        public int getBalance() {                   return balance;       }
        public void setBalance(int balance) {               this.balance = balance;  }

        public static void main(String[] args) {
                Account obj = new Account();
                obj.setName("Tom");
                obj.setBalance(100000);
                System.out.println("이름은 " + obj.getName() + " 통장 잔고는 "
                                + obj.getBalance() + "입니다.");
        }
}
```

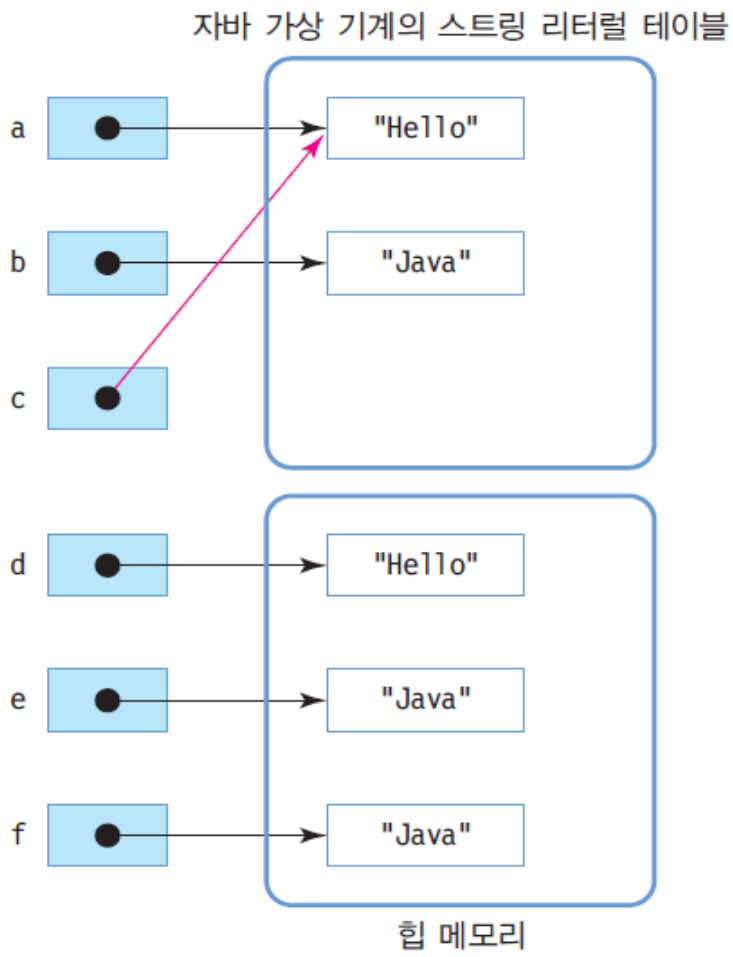
실행결과

이름은 Tom 통장 잔고는 100000입니다.

String 클래스 메소드

char	charAt(int index) 지정된 인덱스에 있는 문자를 반환한다.
int	compareTo(String anotherString) 사전적 순서로 문자열을 비교한다.앞에 있으면 −1, 같으면 0, 뒤에 있으면 1이 반환된다.
String	concat(String str) 주어진 문자열을 현재의 문자열 뒤에 붙인다.
boolean	equals(Object anObject) 주어진 객체와 현재의 문자열을 비교한다.
boolean	equalsIgnoreCase(String anotherString) 대소문자를 무시하고 비교한다.
boolean	isEmpty() length()가 0이면 true를 반환한다.
int	length() 현재 문자열의 길이를 반환한다.
String	replace(char oldChar, char newChar) 주어진 문자열에서 oldChar를 newChar로 변경한, 새로운 문자열을 생성하여 반환한다.
String	substring(int beginIndex, int endIndex) 현재 문자열의 일부를 반환한다.
String	toLowerCase() 문자열의 문자들을 모두 소문자로 변경한다.
String	toUpperCase() 문자열의 문자들을 모두 대문자로 변경한다.

```
String a = "Hello";
String b = "Java";
String c = "Hello";
String d = new String("Hello");
String e = new String("Java");
String f = new String("Java");
```



String 객체 생성

```
public class StringTest {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        String s1 = new String("Java");
        String s2 = new String("Java");
        String s3 = "Java";
        String s4 = "Java";
    }
}
```

- charAt()

```
String s = “Hello World!”;
char c = s.charAt(0);   //H가 반환됨
```

- equals()

```
String s1 = “Apple” ;
String s2 = “Apple” ;
String s3 = new String (“Apple”);
System.out.println(s1.equals(s2));   //true
System.out.println(s2.equals(s3));    //true
```

- 문자열에서 == 연산자는 내용비교가 아닌 객체의 주소를 비교

```
String s1 = “Apple” ;
String s2 = “Apple” ;
String s3 = new String (“Apple”);
System.out.println(s1 == s2);   //true
System.out.println(s1 == s3);    //false
```

- indexOf()

```
String s = “The cat is on the table”;
int index = s.indexOf(“table”);


if (index == -1)
   System.out.println(“table은 없습니다.”);
else
   System.out.println(“table의 위치 : “ + index);    // index로 18이 출력
```

- int compareTo(String anotherString)
  - 문자열이 같으면 0 리턴
  - 이 문자열이 anotherString 보다 사전에 먼저 나오면 음수 리턴
  - 이 문자열이 anotherString 보다 사전에 나중에 나오면 양수 리턴

```
String a = "java";
String b = "jasa";
int res = a.compareTo(b);
if(res == 0)
   System.out.println("the same");
else if(res < 0)
   System.out.println(a +"<"+b);
else
   System.out.println(a +">"+b);
```

"java" 가 "jasa" 보다 사전에 나중에 나오기 때문에 양수 리턴

```
java>jasa
```