

Lab. BankAccount

은행 계좌를 나타내는 BankAccount 클래스를 만들고 다양한 실험을 하여 보자. BankAccount는 잔고를 나타내는 정수형 멤버 변수(balance)를 가지고 있고 예금 인출 메소드(withdraw)와 예입 메소드(deposit), 현재의 잔고를 반환하는 메소드(getBalance)를 가지고 있다.

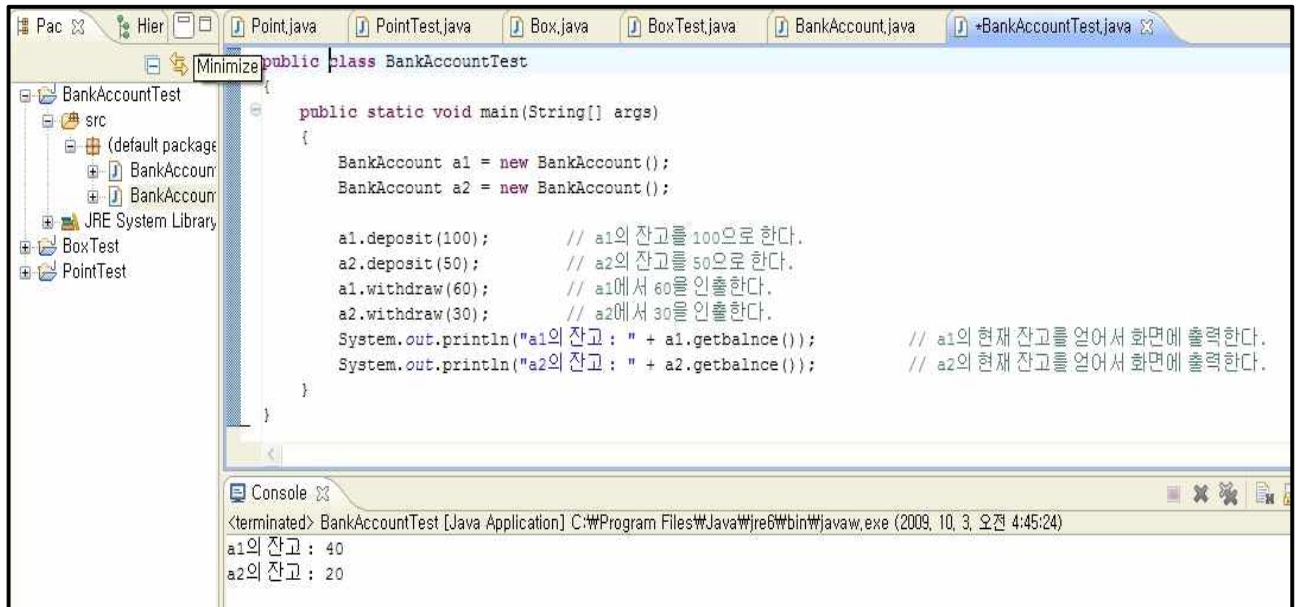
```
class BankAccount                                // 은행 계좌
{
    int balance;                                  // 잔액을 표시하는 변수
    public deposit(int amount)                   // 저금
    {
        _____;
    }
    void withdraw(int amount)                    // 인출
    {
        _____;
    }
    int getbalnce()
    {
        _____;
    }
}
public class BankAccountTest
{
    public static void main(String[] args)
    {
        _____;
    }
}
```

1. 설정자 메소드와 접근자 메소드를 정의하라.

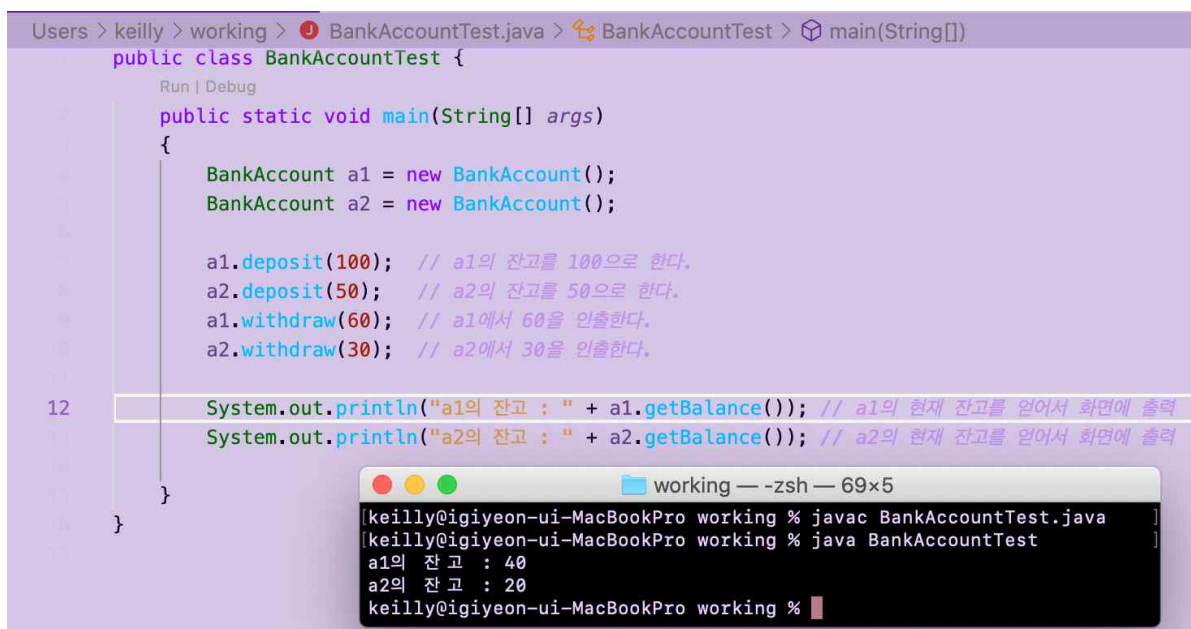
```
BankAccount.java > BankAccount > getBalance()
1  class BankAccount {
2      int balance;
3      public void deposit(int amount)
4      {
5          balance += amount;
6      }
7      void withdraw(int amount)
8      {
9          if (balance < amount)
10         {
11             System.out.println("0");
12         }
13         balance -= amount;
14     }
15     int getBalance()
16     {
17         return balance;
18     }
19 }
```

2. main() 메소드 안에서 a1, a2 두 개의 BankAccount 객체를 생성하고 다음과 같은 순서로 메소드를 호출한다. 잘 수행되는가를 확인하시오.

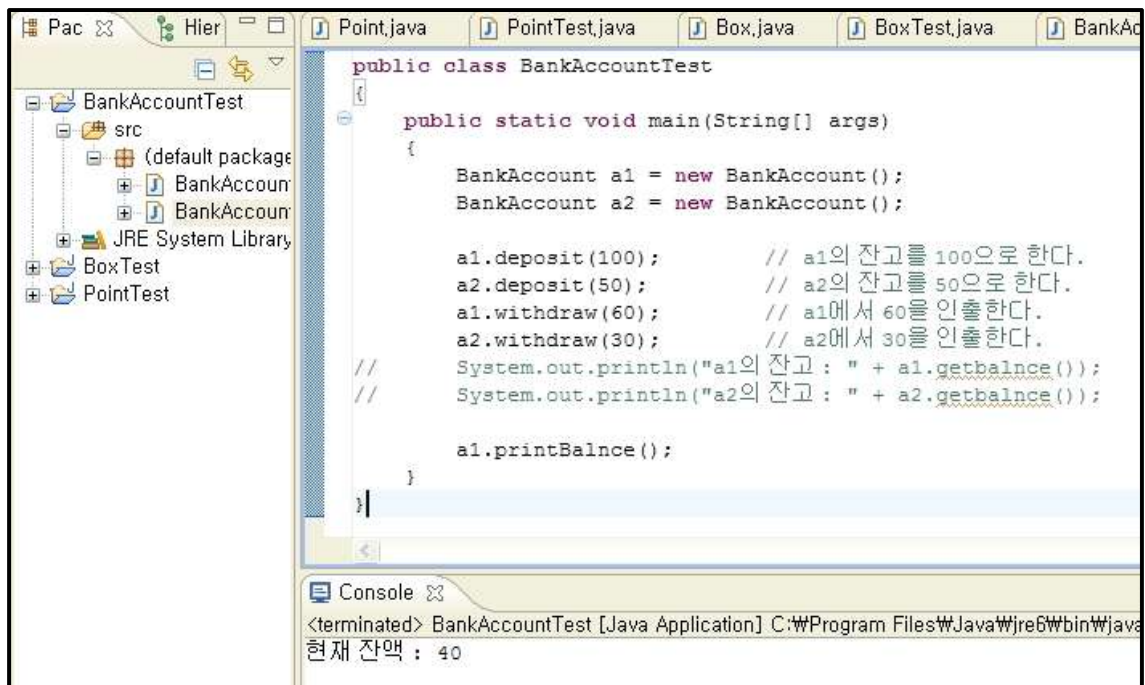
프로그래밍



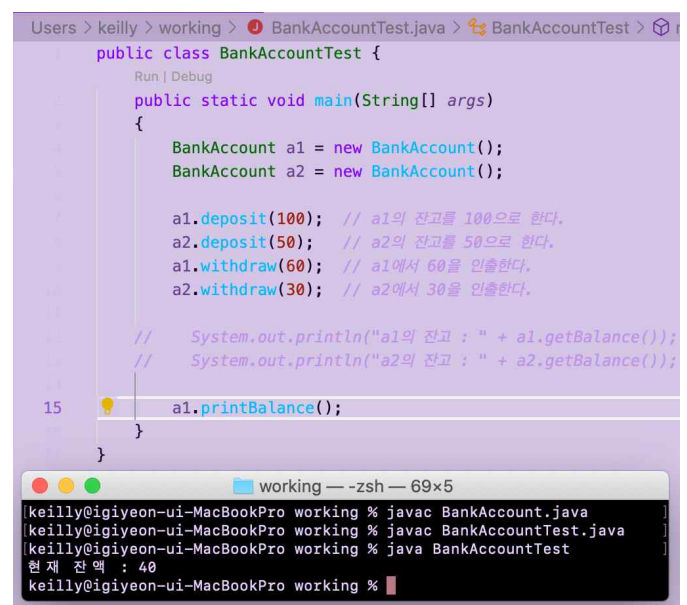
각각의 객체를 생성하여 준 뒤 설정자와 접근자 메소드를 호출하여 준다.



3. 현재 잔액을 출력하는 printBalance() 메소드를 구현하고 테스트하라.

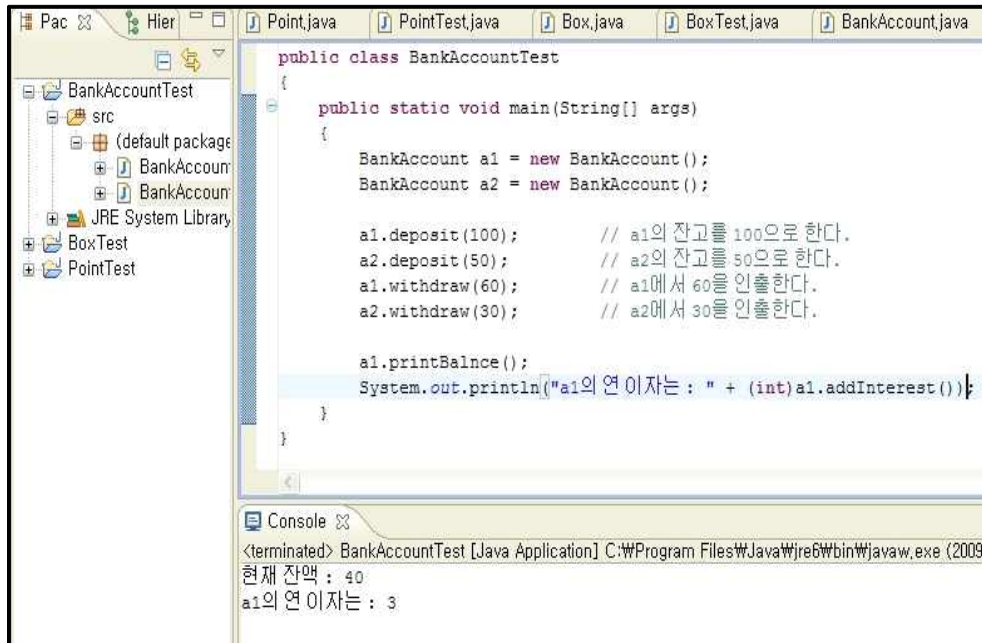


BankAccount 클래스에서 잔액을 출력하는 함수를 선언하여 준 뒤 접근연산자를 이용하여 메소드를 호출하여 주면 된다.



4. 현재 잔액에 대하여 연 7.5%의 이자를 계산하여 추가하는 addInterest() 메소드를 구현하고 테스트하라.

프로그래밍

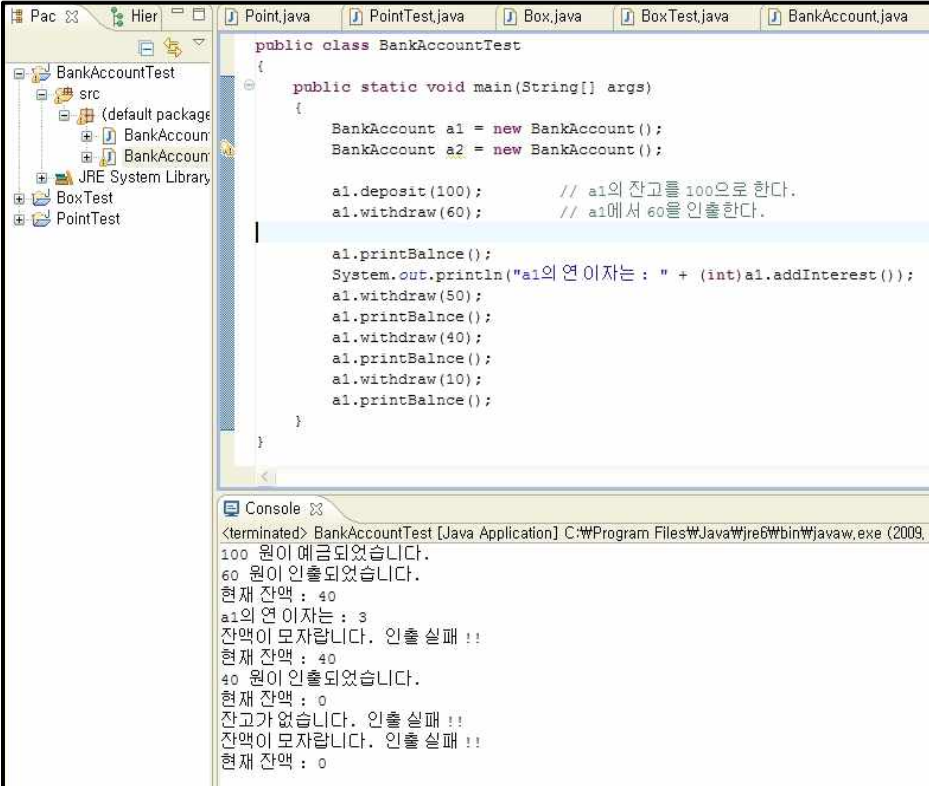


연 이자 7.5%는 현재 예금된 잔액에 0.075를 곱하여 주면 얻을 수 있다.



5. BankAccount 클래스의 예금 인출 메소드인 withdraw()를 현재 잔고가 음수이면 예금 인출이 일어나지 않도록 변경하라. 이러한 BankAccount 클래스의 변경으로 BankAccountTest 클래스를 변경하여야 하는가?

프로그래밍



```
public class BankAccountTest
{
    public static void main(String[] args)
    {
        BankAccount a1 = new BankAccount();
        BankAccount a2 = new BankAccount();

        a1.deposit(100);    // a1의 잔고를 100으로 한다.
        a1.withdraw(60);    // a1에서 60을 인출한다.

        a1.printBalnce();
        System.out.println("a1의 연 이자는 : " + (int)a1.addInterest());
        a1.withdraw(50);
        a1.printBalnce();
        a1.withdraw(40);
        a1.printBalnce();
        a1.withdraw(10);
        a1.printBalnce();
    }
}
```

Console Output:

```
<terminated> BankAccountTest [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (2009.1.1)
100 원이 예금되었습니다.
60 원이 인출되었습니다.
현재 잔액 : 40
a1의 연 이자는 : 3
잔액이 모자랍니다. 인출 실패 !!
현재 잔액 : 40
40 원이 인출되었습니다.
현재 잔액 : 0
잔고가 없습니다. 인출 실패 !!
잔액이 모자랍니다. 인출 실패 !!
현재 잔액 : 0
```

예금이 인출하려는 금액보다 작으면 인출되지 않아야 하며, 또한 예금이 0원이거나 음수일 때(대출) 인출이 되지 않아야 한다. 이러한 것들을 withdraw() 메소드에 적용시켜 주었으며 BankAccount클래스를 변경하였다고 하여 BankAccountTest 클래스의 내용은 변경할 일이 없다. BankAccountTest 클래스에서는 BankAccount 객체를 생성하고 변경된 메소드를 호출하는 일만 하기 때문이다.


```

BankAccount.java > BankAccount > addInterest()
class BankAccount {
    int balance;
    public void deposit(int amount)
    {
        balance += amount;
        System.out.println(amount + " 원이 예금되었습니다.");
    }
    void withdraw(int amount)
    {
        if (balance < amount)
        {
            if (balance == 0)
            {
                System.out.println("잔고가 없습니다. 인출 실패!!");
            }
            System.out.println("잔액이 모자랍니다. 인출 실패!!");
        }
        else
        {
            balance -= amount;
            System.out.println(amount + " 원이 인출되었습니다.");
        }
    }
    void printBalance()
    {
        System.out.println("현재 잔액 : " + balance);
    }
    float addInterest()
    {
        float balance1;
        balance1 = (float)(balance * 0.075);
        return balance1;
    }
}

```

```

Users > keilly > working > BankAccountTest.java > BankAccountTest > main(String[])
public class BankAccountTest {
    Run | Debug
    public static void main(String[] args)
    {
        BankAccount a1 = new BankAccount();
        BankAccount a2 = new BankAccount();

        a1.deposit(100); // a1의 잔고를 100으로 한다.
        a1.withdraw(60); // a1에서 60을 인출한다.
        a1.printBalance(); // 잔액 출력
        System.out.println("a1의 연 이자는 : " + (int)a1.addInterest());
        a1.withdraw(50); // a1에서 50을 인출한다.
        a1.printBalance();
        a1.withdraw(40); // a1에서 40을 인출한다.
        a1.printBalance();
        a1.withdraw(10); // a1에서 10을 인출한다.
        a1.printBalance();
    }
}

```

working — zsh — 69x12

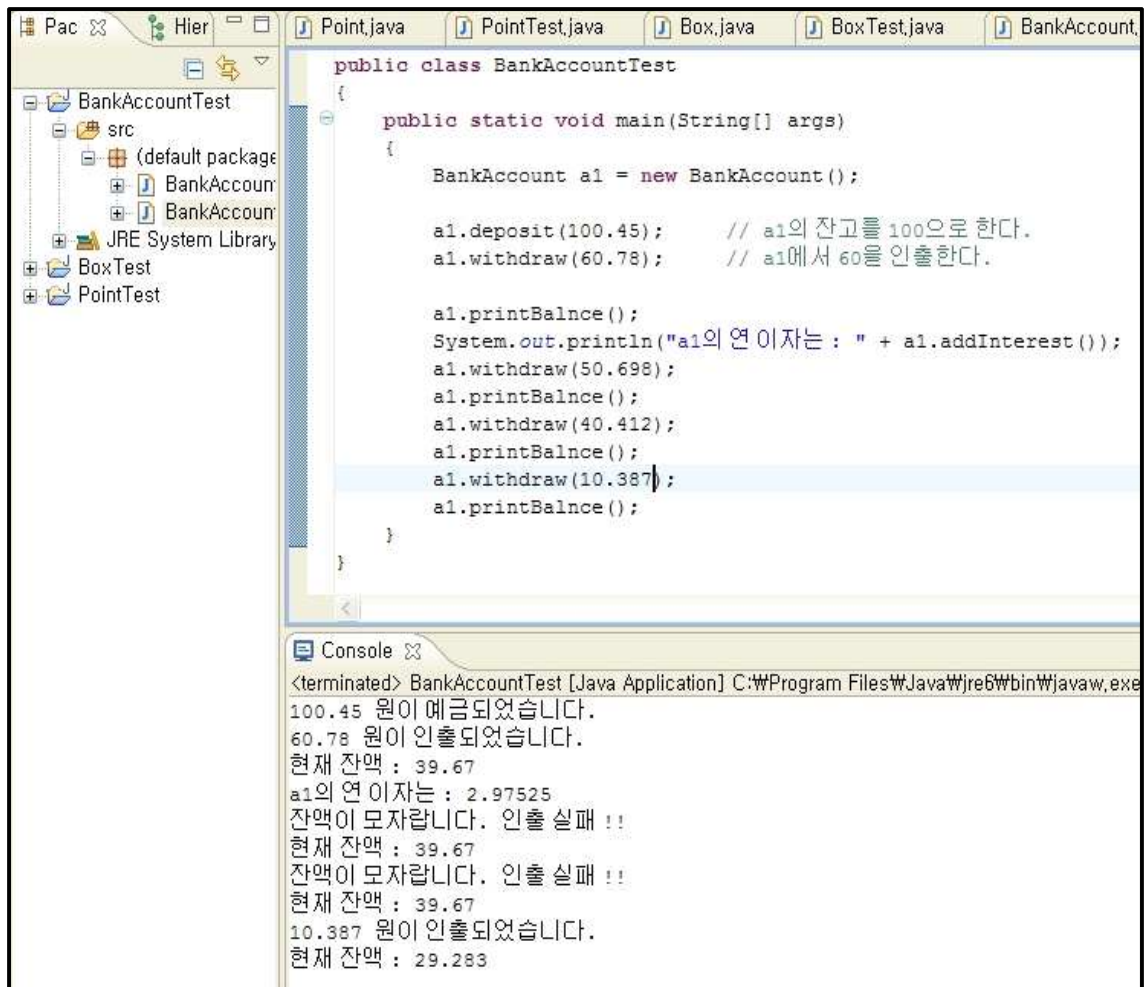
```

[keilly@igiyeeon-ui-MacBookPro working % java BankAccountTest
100 원이 예금되었습니다.
60 원이 인출되었습니다.
현재 잔액 : 40
a1의 연 이자는 : 3
잔액이 모자랍니다. 인출 실패!!
현재 잔액 : 40
40 원이 인출되었습니다.
현재 잔액 : 0
잔고가 없습니다. 인출 실패!!
잔액이 모자랍니다. 인출 실패!!
현재 잔액 : 0

```

6. BankAccount 클래스의 잔고를 나타내는 balance 멤버 변수의 자료형을 int형에서 double 형으로 변경하라. BankAccount 클래스의 변경으로 BankAccountTest클래스를 변경하여야 하는가?

프로그래밍



The screenshot shows an IDE with the following components:

- Project Explorer (Left):** Shows a project named 'BankAccountTest' with a 'src' folder containing 'BankAccount.java' and 'BankAccountTest.java'. It also shows the 'JRE System Library' and other test classes like 'BoxTest' and 'PointTest'.
- Editor (Center):** Displays the code for 'BankAccountTest.java'. The code is as follows:

```
public class BankAccountTest
{
    public static void main(String[] args)
    {
        BankAccount a1 = new BankAccount();

        a1.deposit(100.45);    // a1의 잔고를 100으로 한다.
        a1.withdraw(60.78);    // a1에서 60을 인출한다.

        a1.printBalnce();
        System.out.println("a1의 연 이자는 : " + a1.addInterest());
        a1.withdraw(50.698);
        a1.printBalnce();
        a1.withdraw(40.412);
        a1.printBalnce();
        a1.withdraw(10.387);
        a1.printBalnce();
    }
}
```
- Console (Bottom):** Shows the output of the program execution:

```
<terminated> BankAccountTest [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe
100.45 원이 예금되었습니다.
60.78 원이 인출되었습니다.
현재 잔액 : 39.67
a1의 연 이자는 : 2.97525
잔액이 모자랍니다. 인출 실패 !!
현재 잔액 : 39.67
잔액이 모자랍니다. 인출 실패 !!
현재 잔액 : 39.67
10.387 원이 인출되었습니다.
현재 잔액 : 29.283
```

BankAccountTest 클래스는 BankAccount클래스의 메소드를 호출하여 이용하기 때문에, BankAccountTest의 클래스는 굳이 변경할 필요는 없다.


```

1 class BankAccount {
2     double balance;
3     public void deposit(double amount)
4     {
5         balance += amount;
6         System.out.println(amount + " 원이 예금되었습니다.");
7     }
8     void withdraw(double amount)
9     {
10        if (balance < amount)
11        {
12            if (balance == 0)
13            {
14                System.out.println("잔고가 없습니다. 인출 실패!!");
15            }
16            System.out.println("잔액이 모자랍니다. 인출 실패!!");
17        }
18        else
19        {
20            balance -= amount;
21            System.out.println(amount + " 원이 인출되었습니다.");
22        }
23    }
24    void printBalance()
25    {
26        System.out.println("현재 잔액 : " + balance);
27    }
28    double addInterest()
29    {
30        double balance1;
31        balance1 = (double)(balance * 0.075);
32        return balance1;
33    }
34 }

```

```

Users > keilly > working > BankAccountTest.java > BankAccountTest > main(String
1     public static void main(String[] args)
2     {
3         BankAccount a1 = new BankAccount();
4
5         a1.deposit(100.45); // a1의 잔고를 100.45으로 한다.
6         a1.withdraw(60.78); // a1에서 60.78을 인출한다.
7
8         a1.printBalance(); // 잔액 출력
9
10        System.out.println("a1의 연 이자는 : " + a1.addInterest());
11        a1.withdraw(50.698); // a1에서 50.698을 인출한다.
12        a1.printBalance();
13        a1.withdraw(40.412); //a1에서 40.412을 인출한다.
14        a1.printBalance();
15        a1.withdraw(10.387); // a1에서 10.387을 인출한다.
16        a1.printBalance();
17    }
18 }

```

working — zsh — 65x11

```

[keilly@igiyen-ui-MacBookPro working % java BankAccountTest
100.45 원이 예금되었습니다.
60.78 원이 인출되었습니다.
현재 잔액 : 39.67
a1의 연 이자는 : 2.97525
잔액이 모자랍니다. 인출 실패!!
현재 잔액 : 39.67
잔액이 모자랍니다. 인출 실패!!
현재 잔액 : 39.67
10.387 원이 인출되었습니다.
현재 잔액 : 29.283

```

7. BankAccount 클래스 앞에 public을 추가하고 프로그램을 컴파일 해보자. 어떤 오류가 발생하는가? public을 붙이는 것과 붙이지 않는 것의 차이는 무엇인가?

-> 오류가 발생하지 않았다. public을 붙이면 같은 패키지가 아니더라도 다른 클래스에서 접근할 수 있는 반면, 붙이지 않을 경우 자동으로 default로 선언 후 같은 클래스에서만 가능하도록 한다.

```
BankAccount.java > BankAccount > printBalance()
public class BankAccount {
    double balance;
    public void deposit(double amount)
    {
        balance += amount;
        System.out.println(amount + " 원이 입금되었습니다.");
    }
    void withdraw(double amount)
    {
        if (balance < amount)
        {
            if (balance == 0)
            {
                System.out.println("잔고가 없습니다. 인출 실패!!");
            }
            System.out.println("잔액이 모자랍니다. 인출 실패!!");
        }
        else
        {
            balance -= amount;
            System.out.println(amount + " 원이 인출되었습니다.");
        }
    }
}
24 void printBalance()
{
    System.out.println("현재 잔액 : " + balance);
}
double addInterest()
{
    double balance1;
    balance1 = (double)(balance * 0.075);
    return balance1;
}
```

```
Users > keilly > working > BankAccountTest.java > BankAccountTest > main(String
public static void main(String[] args)
{
    BankAccount a1 = new BankAccount();

    a1.deposit(100.45); // a1의 잔고를 100.45으로 한다.
    a1.withdraw(60.78); // a1에서 60.78을 인출한다.

    a1.printBalance(); // 잔액 출력
    System.out.println("a1의 연 이자는 : " + a1.addInterest());
    a1.withdraw(50.698); // a1에서 50.698을 인출한다.
    a1.printBalance();
    a1.withdraw(40.412); //a1에서 40.412을 인출한다.
    a1.printBalance();
    a1.withdraw(10.387); // a1에서 10.387을 인출한다.
    a1.printBalance();
}

[keilly@igiyeon-ui-MacBookPro working % java BankAccountTest
100.45 원이 입금되었습니다.
60.78 원이 인출되었습니다.
현재 잔액 : 39.67
a1의 연 이자는 : 2.97525
잔액이 모자랍니다. 인출 실패!!
현재 잔액 : 39.67
잔액이 모자랍니다. 인출 실패!!
현재 잔액 : 39.67
10.387 원이 인출되었습니다.
현재 잔액 : 29.283]
```

8. BankAccount 클래스 안에도 public static void main(String[] args){ }을 추가하여 보라. 컴파일 오류가 발생하는가? 만약 컴파일 오류가 발생하지 않으면 BankAccount 클래스와 BankAccountTest 클래스의 main()메소드에 각각 println() 문장을 넣어서 어떤 main() 메소드가 수행되는지를 확인하라.

```
public class BankAccountTest
{
    public static void main(String[] args)
    {
        System.out.println("BankAccountTest 클래스의 메인함수");
        BankAccount a1 = new BankAccount();

        a1.deposit(100.45);    // a1의 잔고를 100으로 한다.
        a1.withdraw(60.78);   // a1에서 60을 인출한다.

        a1.printBalnce();
        System.out.println("a1의 연 이자는 : " + a1.addInterest());
        a1.withdraw(50.698);
        a1.printBalnce();
        a1.withdraw(40.412);
        a1.printBalnce();
        a1.withdraw(10.387);
        a1.printBalnce();
    }
}
```

Console

<terminated> BankAccountTest [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (3)

BankAccountTest 클래스의 메인함수
100.45 원이 예금되었습니다.
60.78 원이 인출되었습니다.
현재 잔액 : 39.67
a1의 연 이자는 : 2.97525
잔액이 모자랍니다. 인출 실패 !!
현재 잔액 : 39.67
잔액이 모자랍니다. 인출 실패 !!
현재 잔액 : 39.67
10.387 원이 인출되었습니다.
현재 잔액 : 29.283

-> BankAccountTest 클래스의 모든 메소드에 오류가 뜨면서 올바르게 실행되지 않는다.

```
BankAccount.java > BankAccount > main(String[])
public class BankAccount {
    public static void main(String[] args)
    {
        double balance;
        void deposit(double amount)
        {
            balance += amount;
            System.out.println(amount + " 원이 예금되었습니다.");
        }
        void withdraw(double amount)
        {
            if (balance < amount)
            {
                if (balance == 0)
                {
                    System.out.println("잔고가 없습니다. 인출");
                }
                System.out.println("잔액이 모자랍니다. 인출");
            }
            else
            {
                balance -= amount;
                System.out.println(amount + " 원이 인출");
            }
        }
        void printBalance()
        {
            System.out.println("현재 잔액 : " + balance);
        }
        double addInterest()
        {
            double balance1;
            balance1 = (double)(balance * 0.075);
            return balance1;
        }
    }
}

Users > keilly > working > BankAccountTest.java > BankAccountTest > main(String[])
public class BankAccountTest {
    public static void main(String[] args)
    {
        BankAccount a1 = new BankAccount();

        a1.deposit(100.45); // a1의 잔고를 100.45으로 한다.
        a1.withdraw(60.78); // a1에서 60.78을 인출한다.

        a1.printBalance();
        System.out.println(a1.printBalance());
        a1.withdraw(50.6);
        a1.printBalance();
        a1.withdraw(40.4);
        a1.printBalance();
        a1.withdraw(10.3);
        a1.printBalance();
    }
}

keilly@igiyeon-ui-MacBookPro working % javac BankAccountTest.java
BankAccount.java:5: error: illegal start of expression
    void deposit(double amount)
    ^
BankAccount.java:5: error: ';' expected
    void deposit(double amount)
    ^
BankAccount.java:5: error: ';' expected
    void deposit(double amount)
    ^
BankAccount.java:10: error: illegal start of expression
    void withdraw(double amount)
    ^
BankAccount.java:10: error: ';' expected
    void withdraw(double amount)
    ^
BankAccount.java:10: error: ';' expected
    void withdraw(double amount)
    ^
BankAccount.java:26: error: illegal start of expression
    void printBalance()
    ^
BankAccount.java:26: error: ';' expected
    void printBalance()
    ^
BankAccount.java:30: error: ';' expected
    double addInterest()
    ^
9 errors
keilly@igiyeon-ui-MacBookPro working %
```

9. 7번 문제와 8번 문제의 실험 결과를 토대로 public 클래스와 main() 메소드에 대하여 어떤 결론을 내릴 수 있는가?

-> public 클래스의 유무에 따라 같은 패키지가 아닌 다른 클래스에 접근할 수 있는지 아닌지를 가릴 수 있다. 따라서 7번 문제에서 오류가 생기지 않은 것은 두 클래스가 같은 패키지 안에 있기 때문임을 유추할 수 있다.

-> 클래스 내부에 main() 메소드가 있다면 우선적으로 실행되어야 함을 의미하며, main() 메소드가 있는 클래스의 메소드를 따온 다른 클래스는 실행되는 데 있어서 오류가 나타난다.