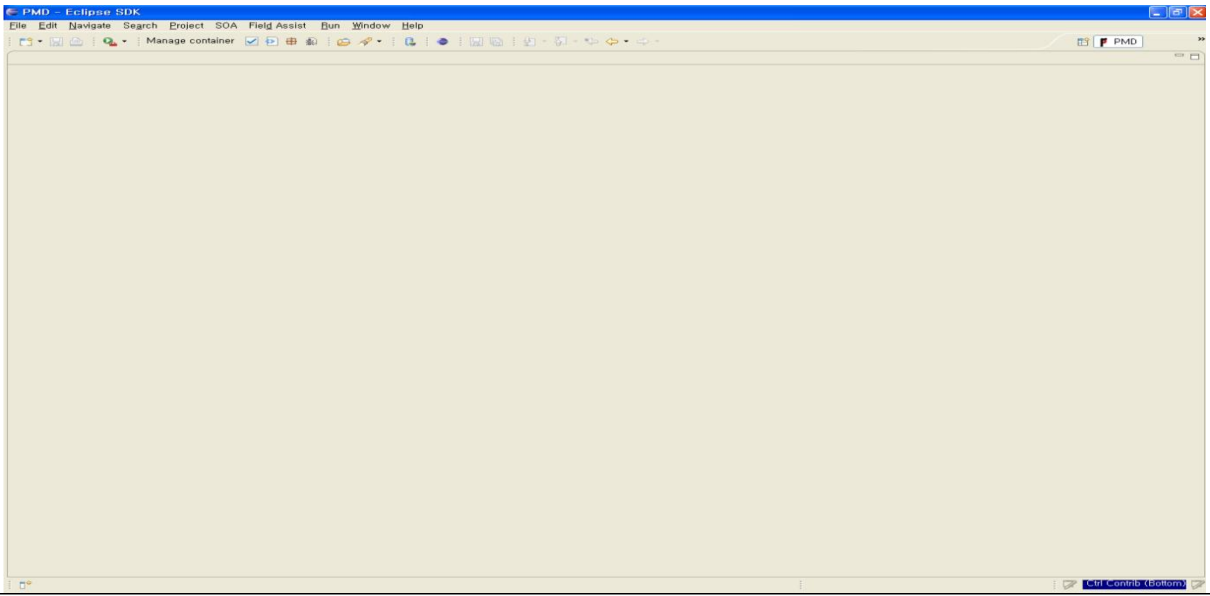
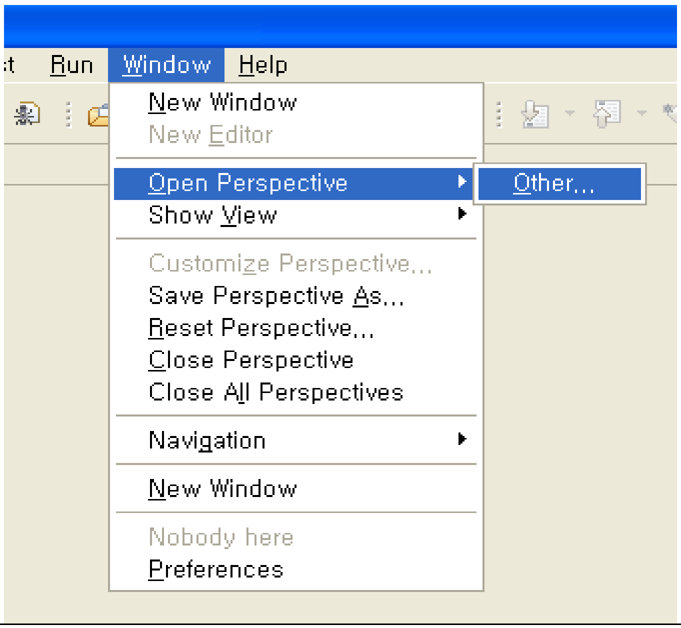
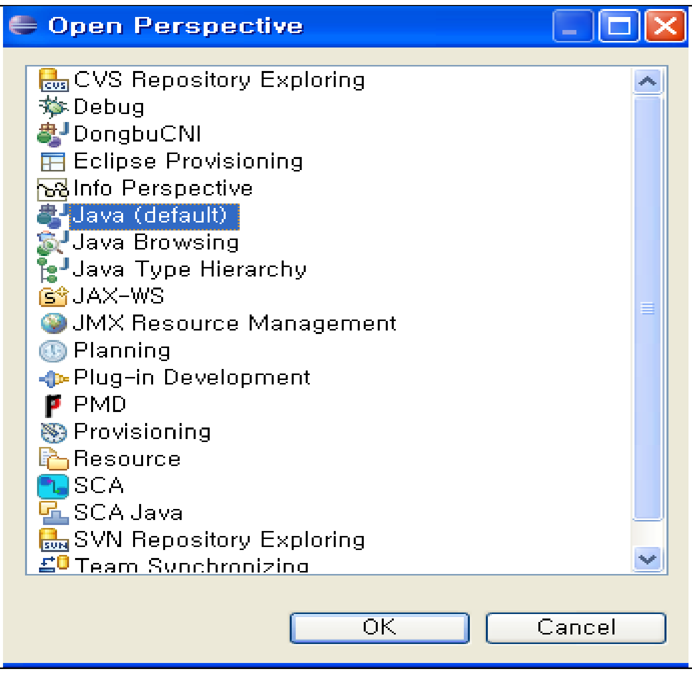
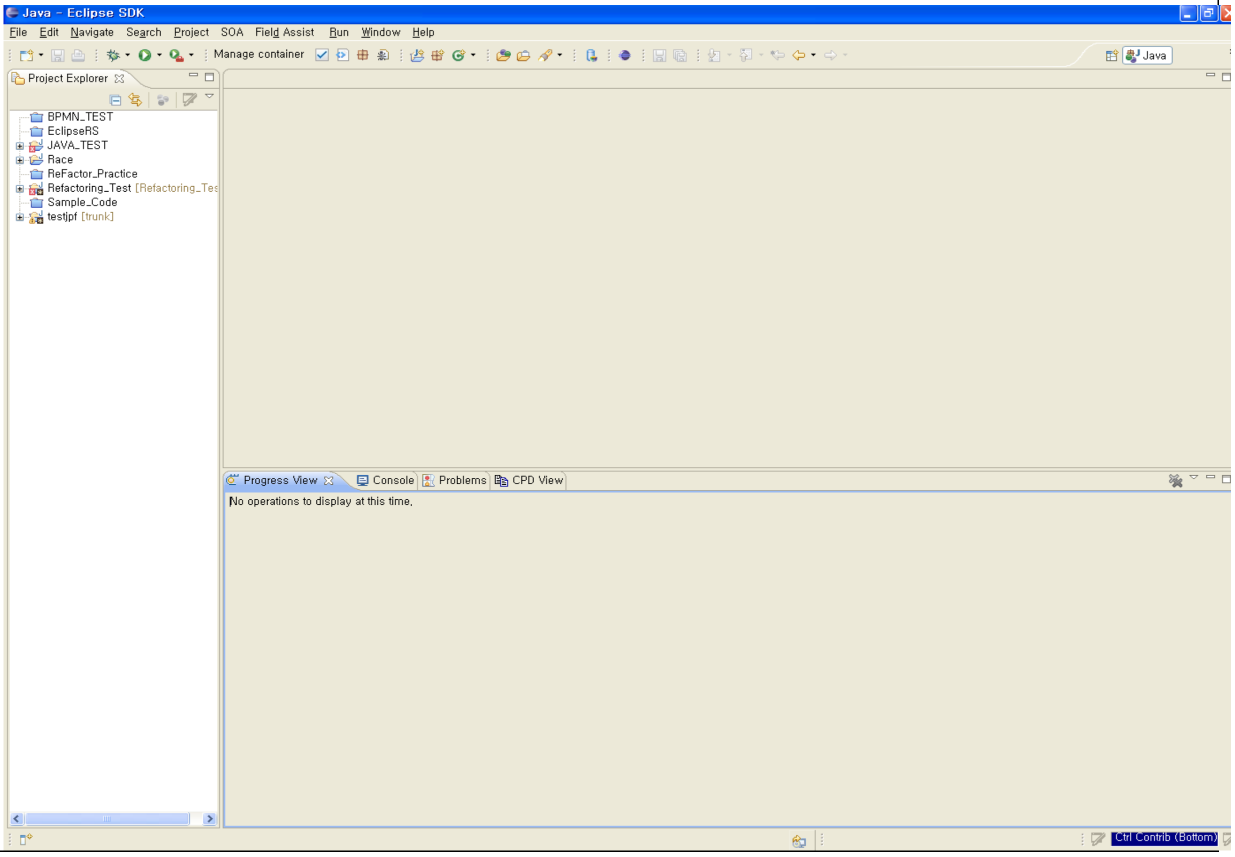
리팩토링 실습

1. 이클립스 화면 셋팅하기

1. 이클립스 툴을 연다.  PMD - Eclipse SDK File Edit Navigate Search Project SOA Field Assist Run Window Help Manage container PMD Ctrl Contrib (Bottom)
2. Window > Open Perspective > Other를 선택한다.  Run Window Help New Window New Editor Open Perspective Other... Show View Customize Perspective... Save Perspective As... Reset Perspective... Close Perspective Close All Perspectives Navigation New Window Nobody here Preferences
3. 대화상자에서 Java(default)를 선택하고 OK를 누른다.



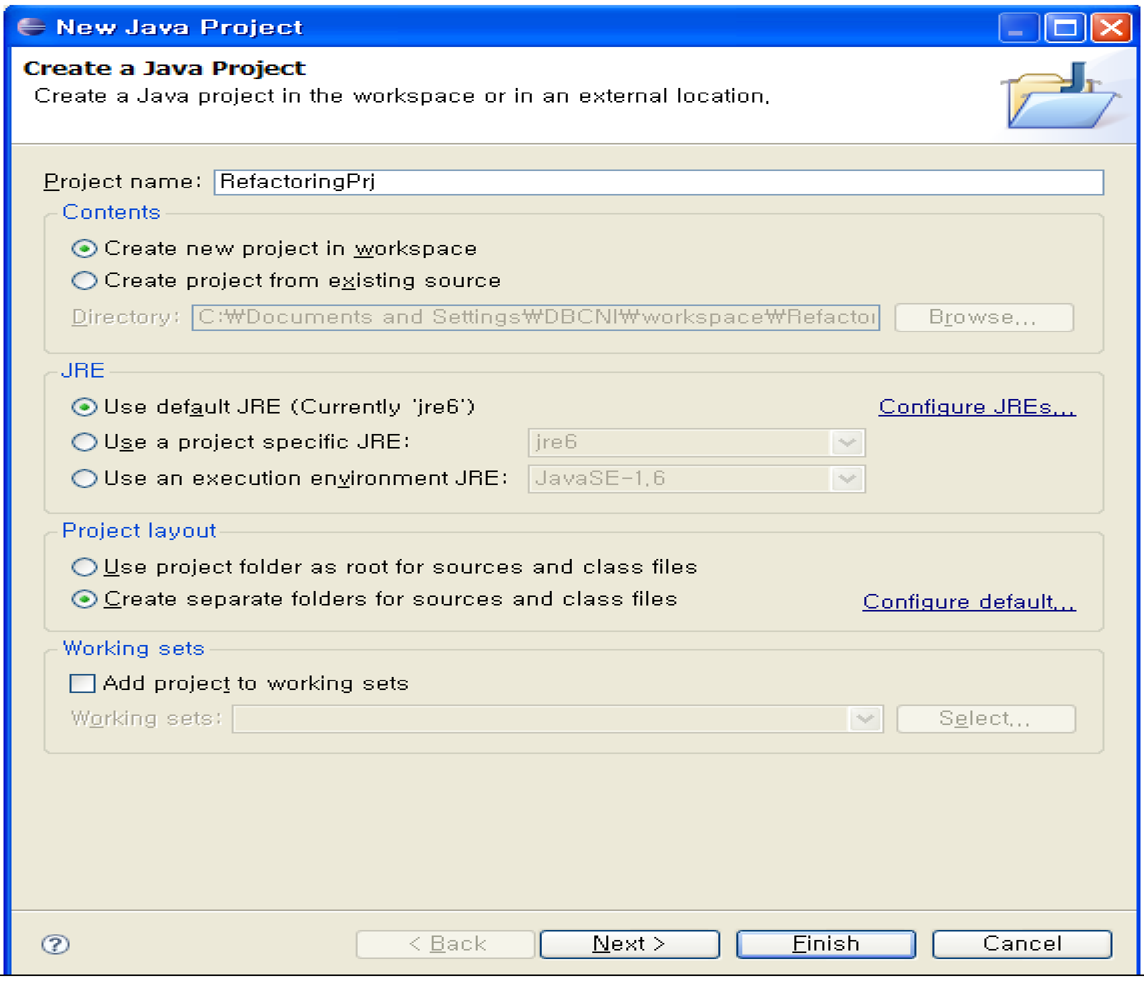
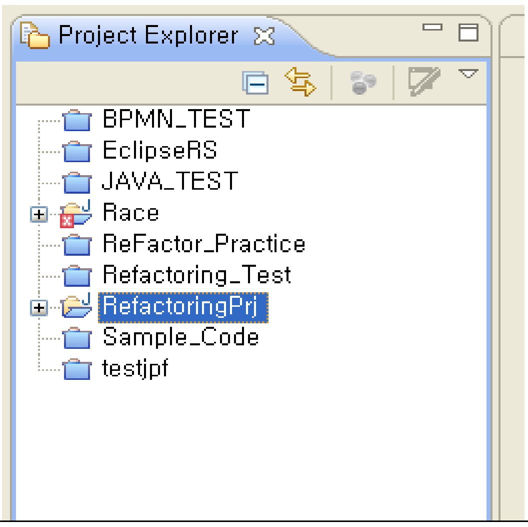
4. 다음과 같은 화면이 나타난다.



No operations to display at this time.

2. Java 프로젝트 만들기

Movie 클래스 만들기

1. File > New > Java Project를 선택한다.
2. 대화상자가 나타나면 프로젝트명에 RefactoringPrj를 넣는다. 
3. Next를 누르고 완료한다. 완료하면 화면이 다음과 같이 바뀐다. 

New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: RefactoringPrj

Contents
- [✓ Create new project in workspace]
- Create project from existing source

Directory: C:\Documents and Settings\DBCNI\workspace\Refactor

JRE
- [✓ Use default JRE (Currently 'jre6')]
- Use a project specific JRE: jre6
- Use an execution environment JRE: JavaSE-1.6

Project layout
- Use project folder as root for sources and class files
- [✓ Create separate folders for sources and class files]

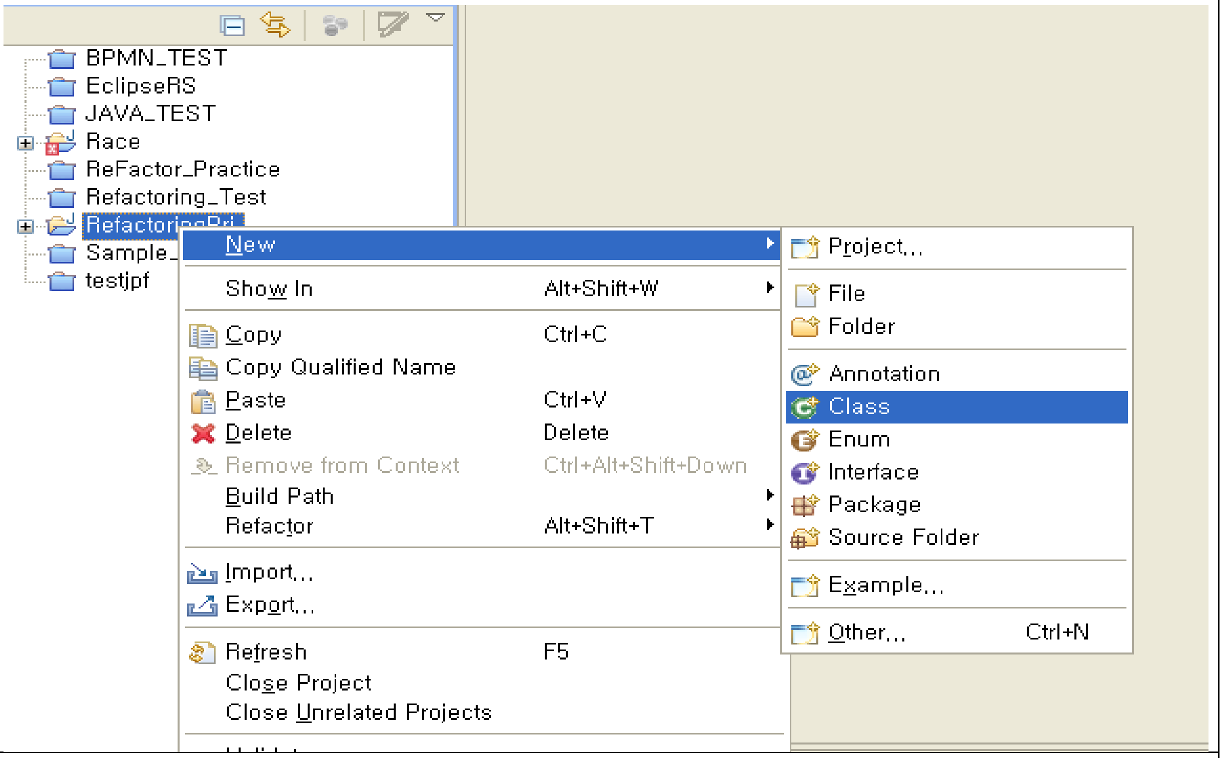
Working sets
- Add project to working sets

Working sets:

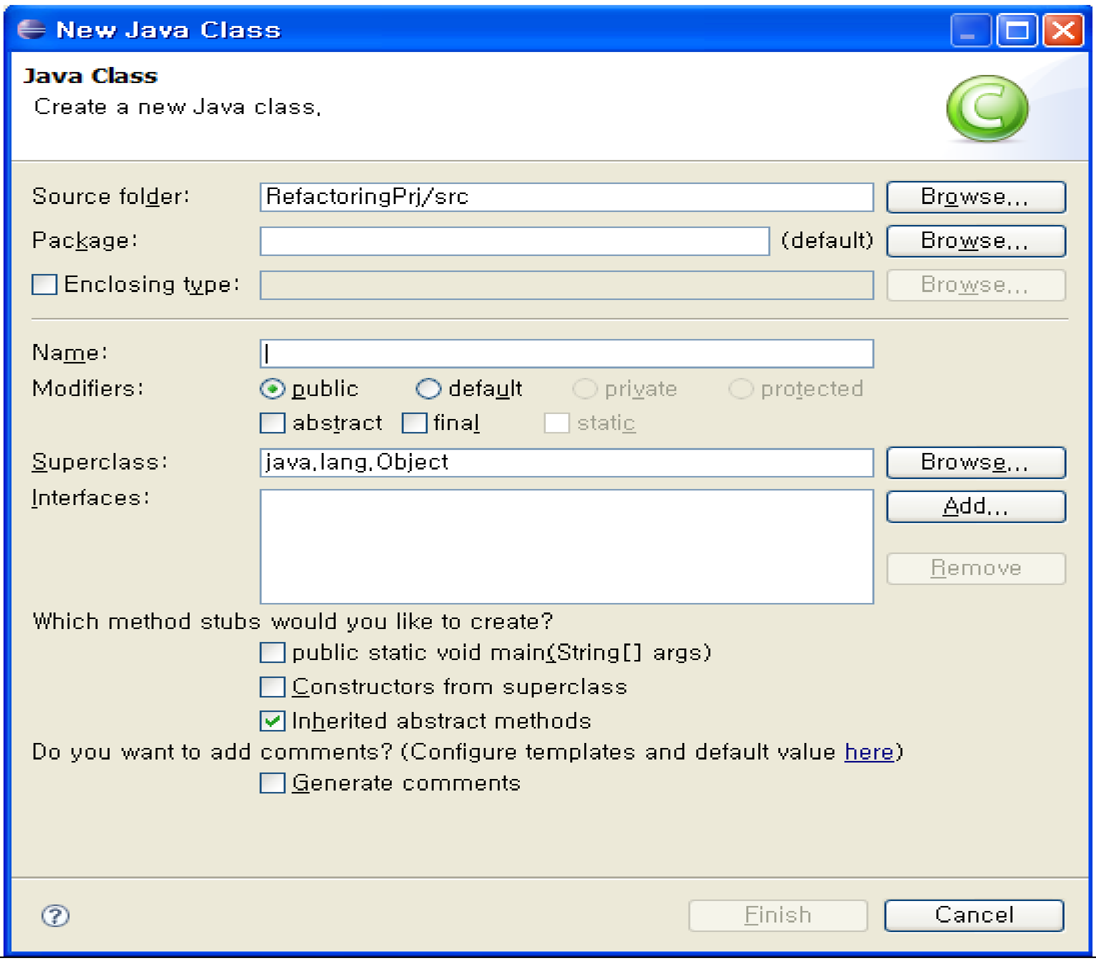
Project Explorer

- BPMN_TEST
- EclipseRS
- JAVA_TEST
- Race
- ReFactor_Practice
- Refactoring_Test
- RefactoringPrj
- Sample_Code
- testjpf

4. 프로젝트를 선택하고 오른쪽 마우스 버튼을 눌러서 New > Class를 선택한다.



5. 다음과 같은 대화상자가 나타난다.


New Java Class

Java Class

Create a new Java class.

Source folder: RefactoringPrj/src [Browse...]

Package: (default) [Browse...]

☐ Enclosing type: [Browse...]

Name: |

Modifiers: [✓ public] ○ default ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object [Browse...]

Interfaces: [Add...] [Remove]

Which method stubs would you like to create?

☐ public static void main(String[] args)

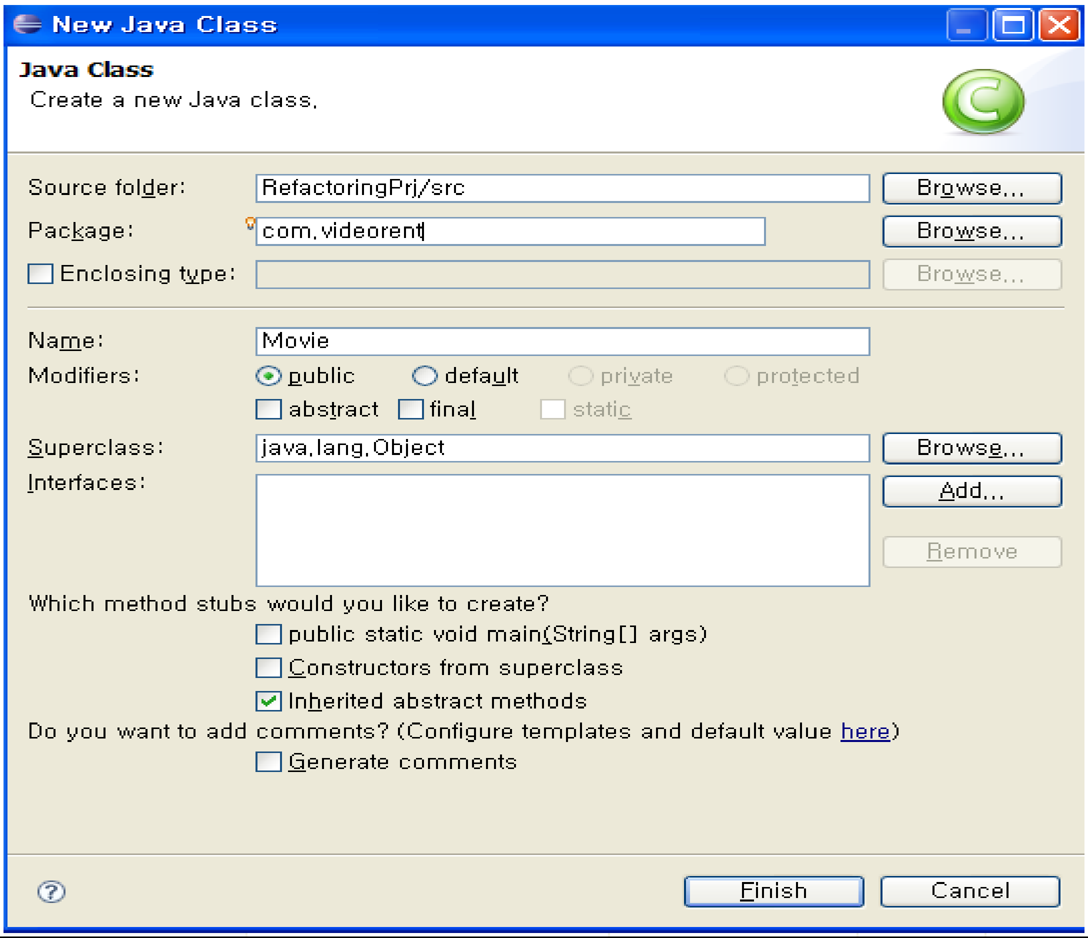
☐ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

[Finish] [Cancel]

6. Package 명은 com.videorent를 입력하고 Name에는 Movie를 입력하고 Finish 버튼을 누른다. 
7.; 화면이 다음과 같이 바뀐다.

New Java Class

Java Class

Create a new Java class.

Source folder: RefactoringPrj/src

Package: com.videorent

☐ Enclosing type:

Name: Movie

Modifiers: [✓ public] default private protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

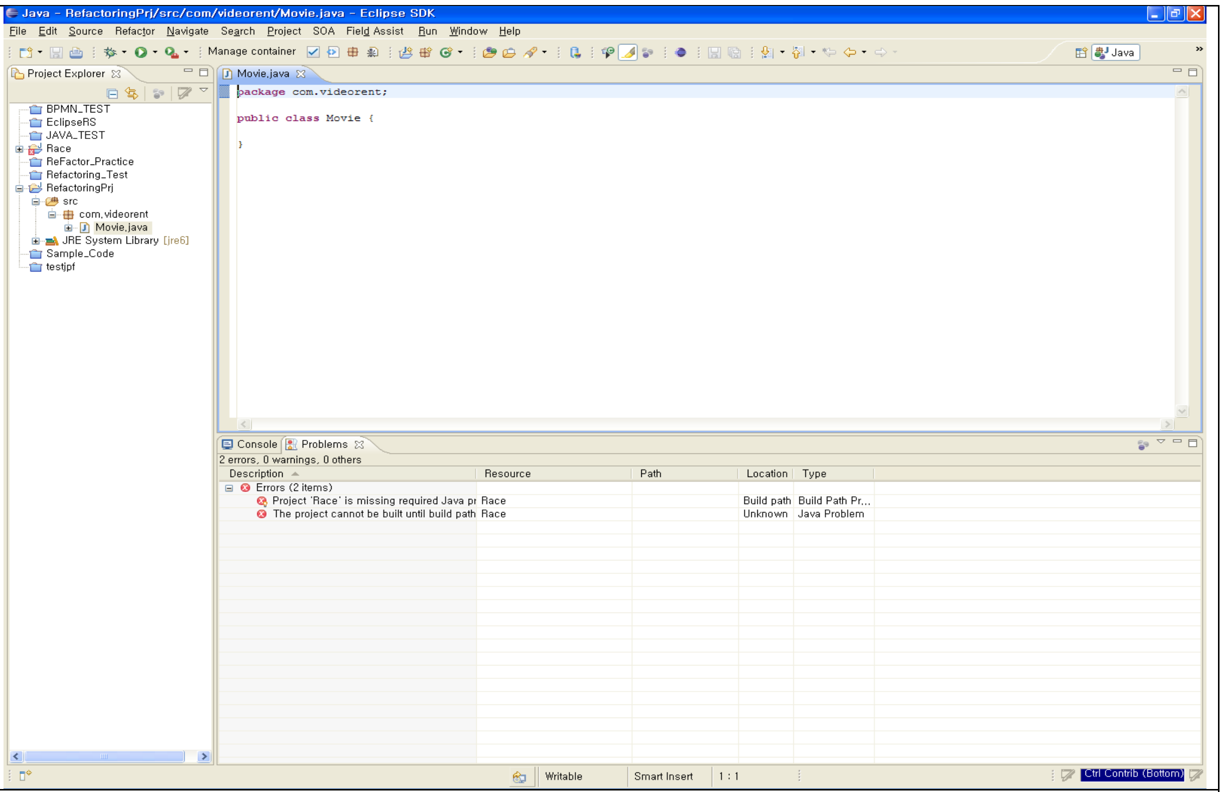
☐ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

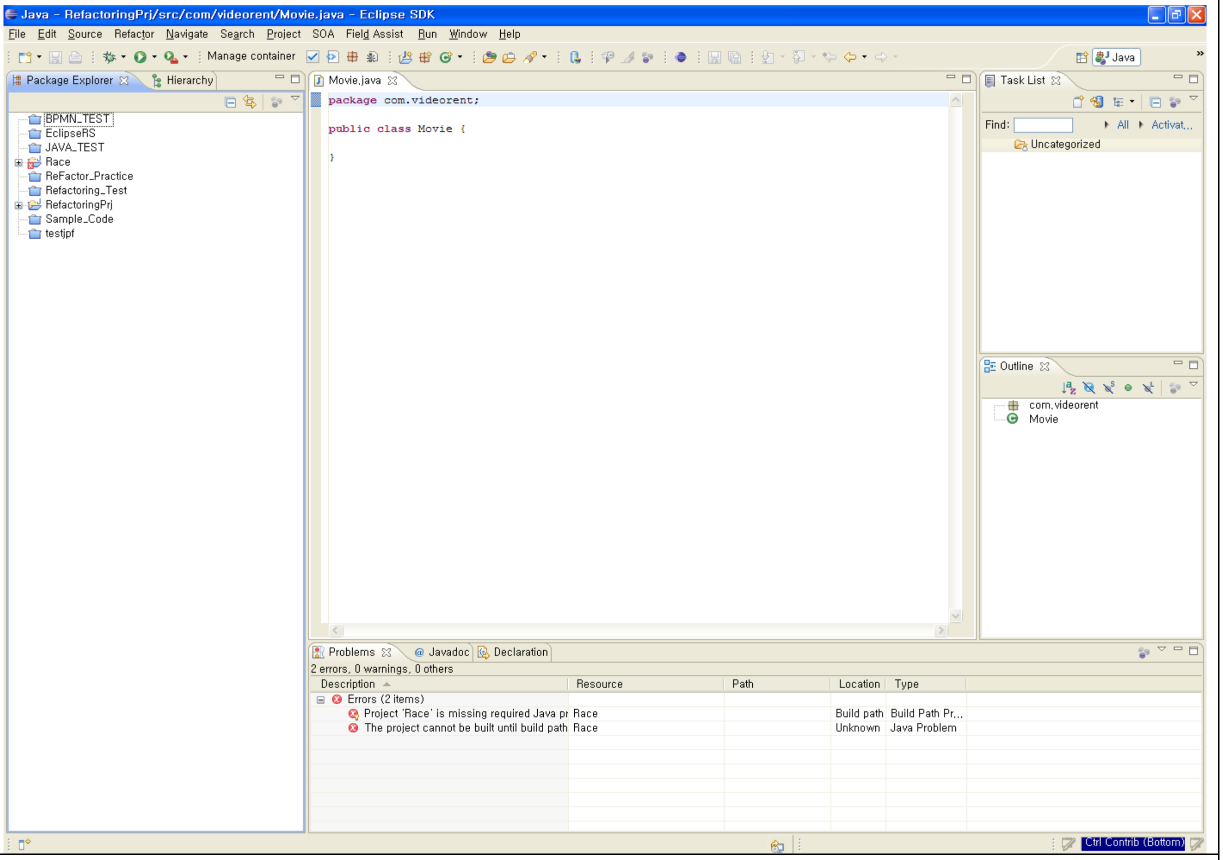
☐ Generate comments



```
package com.videorent;

public class Movie {

}
```

```
2 errors, 0 warnings, 0 others
Description                                  Resource   Path   Location      Type
Errors (2 items)
  Project 'Race' is missing required Java pr  Race              Build path    Build Path Pr...
  The project cannot be built until build path Race             Unknown       Java Problem
```

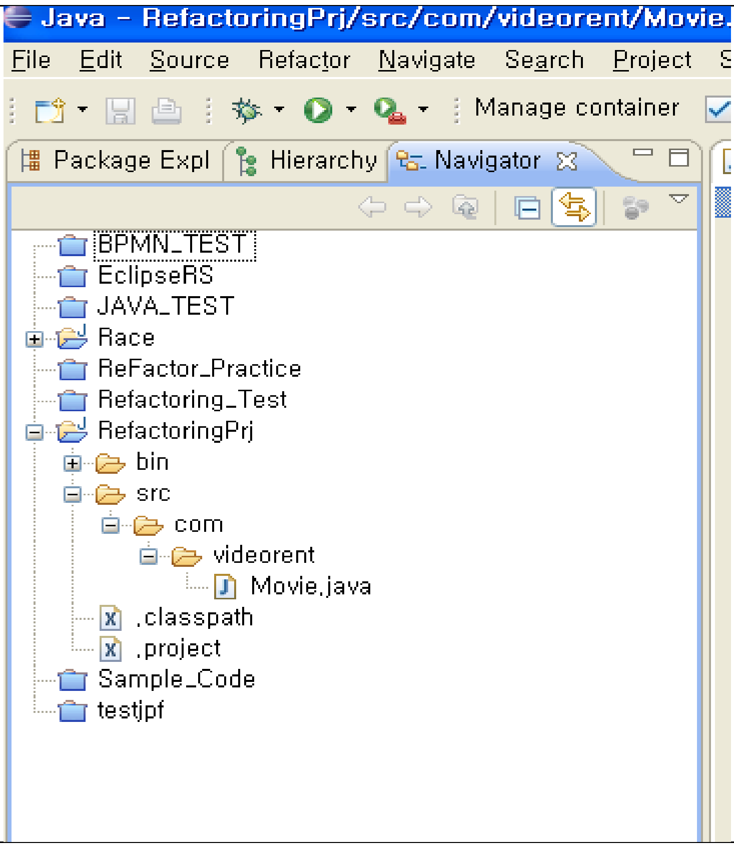
8. Window > Reset Perspective를 누른다. 화면이 아래와 같이 바뀐다.



```
package com.videorent;

public class Movie {

}
```

```
2 errors, 0 warnings, 0 others
Description                                  Resource   Path   Location      Type
Errors (2 items)
  Project 'Race' is missing required Java pr  Race              Build path    Build Path Pr...
  The project cannot be built until build path Race             Unknown       Java Problem
```

9. Window > Show View > Navigator를 선택한다. Navigator View가 나타난다.



10. 다음 문장을 입력한다.

```
package com.videorent;

public class Movie {
public static final int CHILDRENS = 2;
public static final int REGULAR = 0;
public static final int NEW_RELEASE = 1;
}
```

11. Ctrl + Shift + F를 눌러서 포맷을 조정한다.

```
package com.videorent;

public class Movie {
	public static final int CHILDRENS = 2;
	public static final int REGULAR = 0;
	public static final int NEW_RELEASE = 1;
}
```

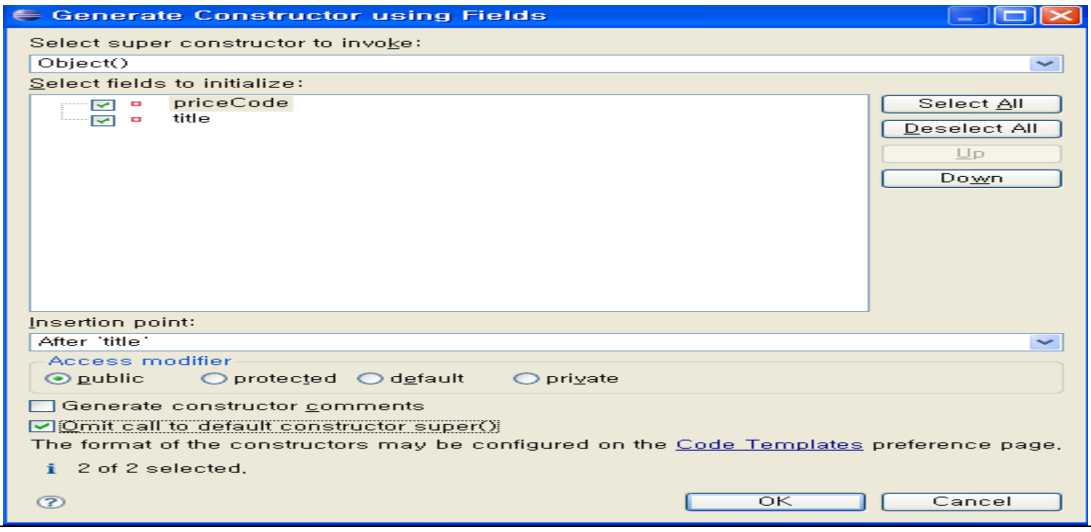
12. 속성을 추가한다.

```
package com.videorent;

public class Movie {

	public static final int CHILDRENS = 2;
```

```
        public static final int REGULAR = 0;
        public static final int NEW_RELEASE = 1;

        private String title;
        private int priceCode;
}
```

13. 생성자를 추가한다. Source > Generate Constructor using Fields를 선택한다.

Omit call to default constructor super()를 체크한다.

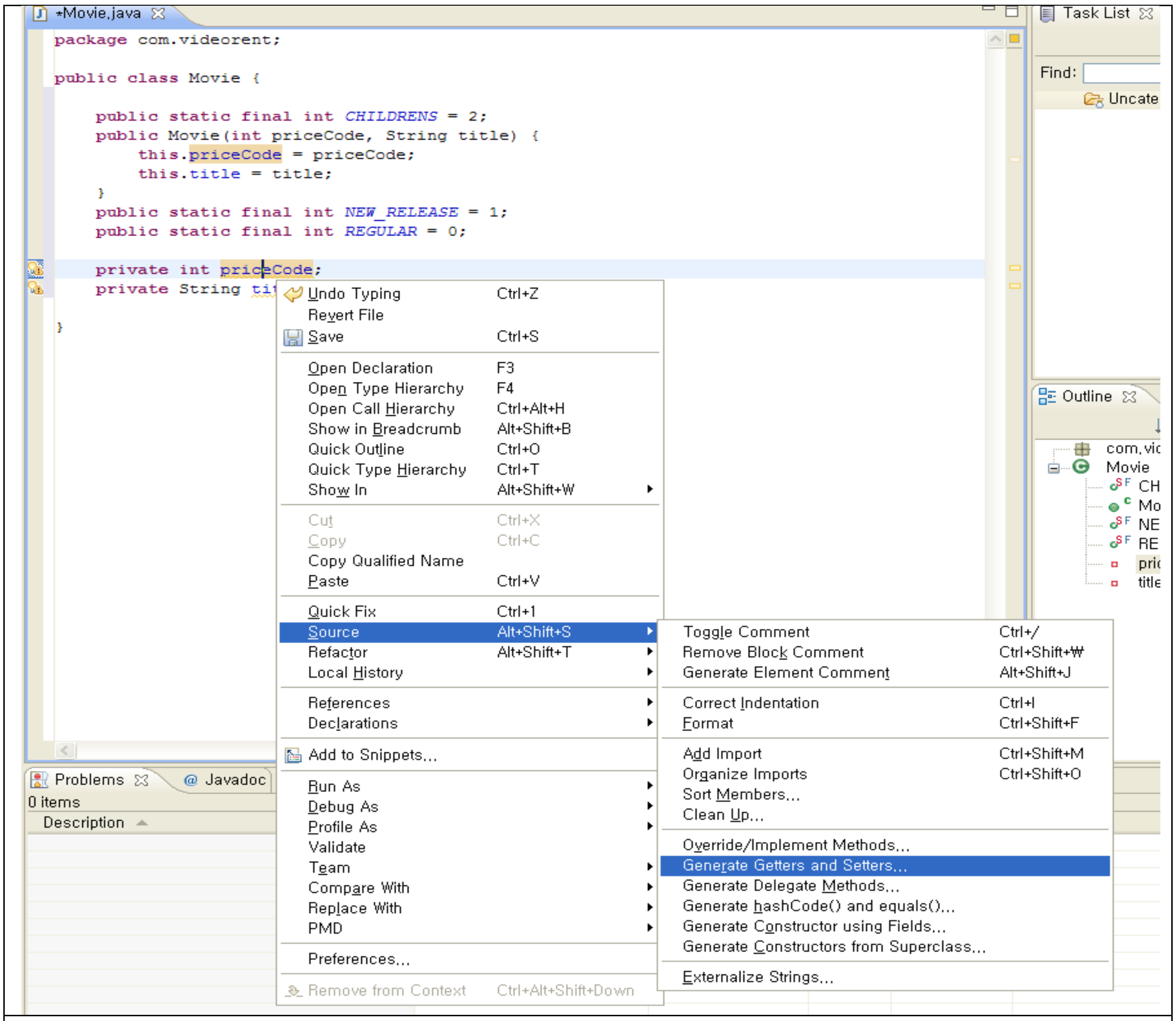

Generate Constructor using Fields
Select super constructor to invoke:
Object()
Select fields to initialize:
priceCode
title
Select All
Deselect All
Up
Down
Insertion point:
After 'title'
Access modifier
public
protected
default
private
Generate constructor comments
Omit call to default constructor super()
The format of the constructors may be configured on the Code Templates preference page.
2 of 2 selected.
OK
Cancel

14. OK를 누르면 코드가 다음과 같이 바뀐다.

```
package com.videorent;

public class Movie {

        public static final int CHILDRENS = 2;
        public Movie(int priceCode, String title) {
                this.priceCode = priceCode;
                this.title = title;
        }
        public static final int NEW_RELEASE = 1;
        public static final int REGULAR = 0;

        private int priceCode;
        private String title;

}
```

15. PriceCode에 대한 get, set 메소드를 만든다.

priceCode를 선택한 후 Source > Generate Getters and Setters를 선택한다.

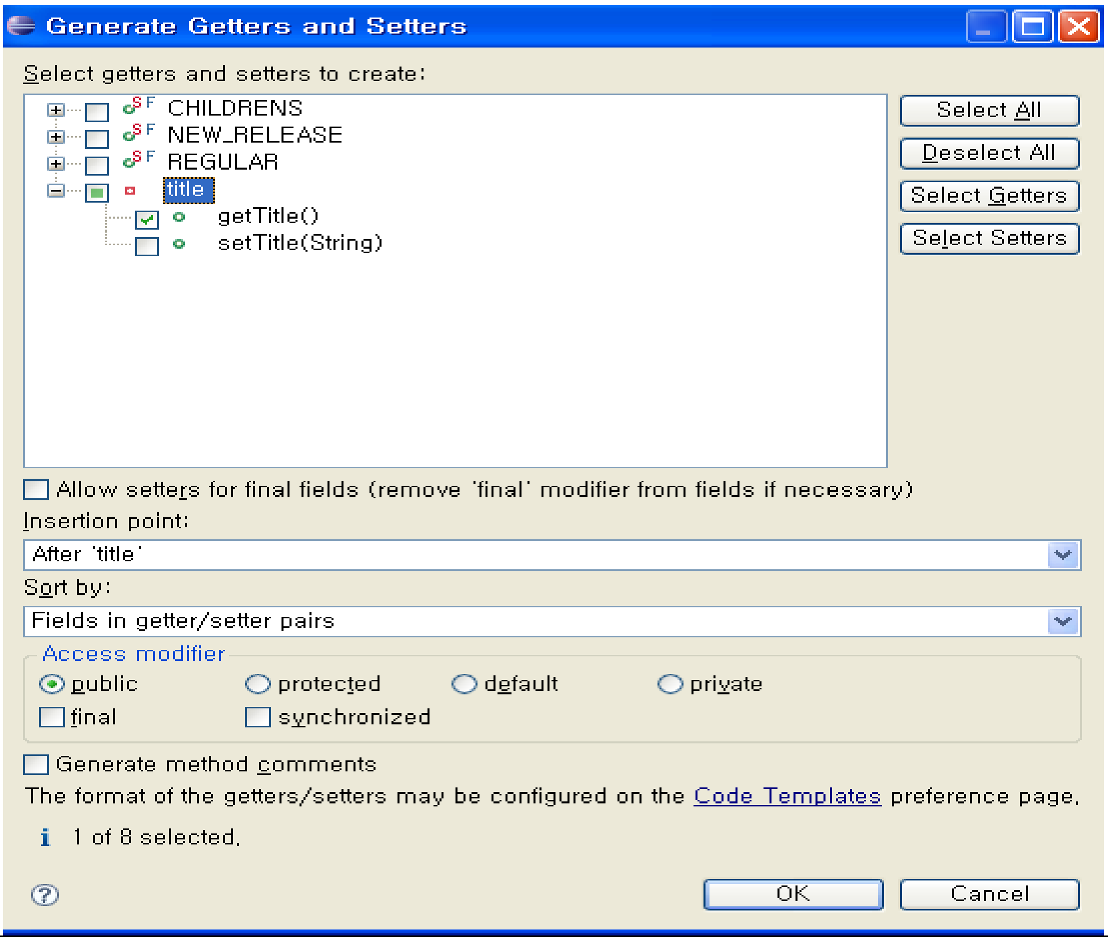


```
package com.videorent;

public class Movie {

    public static final int CHILDRENS = 2;
    public Movie(int priceCode, String title) {
        this.priceCode = priceCode;
        this.title = title;
    }
    public static final int NEW_RELEASE = 1;
    public static final int REGULAR = 0;

    private int priceCode;
    private String tit
}
```

16. 대화상자가 나타나면 OK 버튼을 누른다. 다음과 같은 프로그램이 나타난다.

```
package com.videorent;

public class Movie {

        public static final int CHILDRENS = 2;
        public Movie(int priceCode, String title) {
                this.priceCode = priceCode;
                this.title = title;
        }
        public static final int NEW_RELEASE = 1;
        public static final int REGULAR = 0;

        private int priceCode;
        public int getPriceCode() {
                return priceCode;
        }
        public void setPriceCode(int priceCode) {
                this.priceCode = priceCode;
        }
        private String title;

}
```

17. title 속성에 대해서는 get 메소드만을 만든다. title을 선택한 후 Source > Generate Getters and Setters를 선택한다. 다음과 같은 대화상자가 나타나면 setTitle의 check를 해제한다.



Select getters and setters to create:

- [ ] CHILDRENS
- [ ] NEW_RELEASE
- [ ] REGULAR
- title
  - [✓] getTitle()
  - [ ] setTitle(String)

- [ ] Allow setters for final fields (remove 'final' modifier from fields if necessary)

Insertion point: After 'title'

Sort by: Fields in getter/setter pairs

Access modifier: [✓ public] protected default private

- [ ] final
- [ ] synchronized
- [ ] Generate method comments

The format of the getters/setters may be configured on the Code Templates preference page.

1 of 8 selected.

18. OK 버튼을 누르면 프로그램이 다음과 같이 변형된다.

```
package com.videorent;

public class Movie {

	public static final int CHILDRENS = 2;
	public Movie(int priceCode, String title) {
		this.priceCode = priceCode;
		this.title = title;
	}
	public static final int NEW_RELEASE = 1;
	public static final int REGULAR = 0;

	private int priceCode;
	public int getPriceCode() {
		return priceCode;
	}
	public void setPriceCode(int priceCode) {
		this.priceCode = priceCode;
	}
	private String title;
	public String getTitle() {
		return title;
```

```
        }

}
```

19. 코드를 정리하기 위해 Source > Sort Members를 선택한다. 대화상자가 나타나면 OK를 누른다. 코드가 다음과 같이 변형된다.

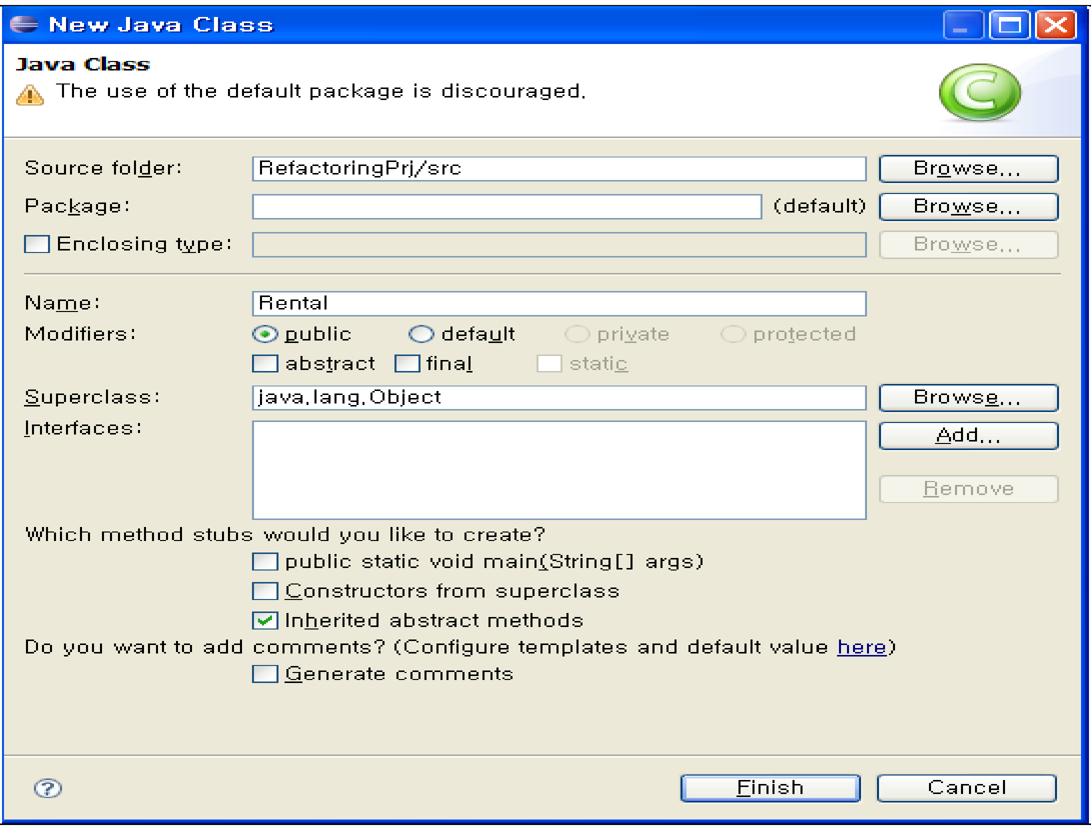
```
package com.videorent;

public class Movie {

        public static final int CHILDRENS = 2;
        public static final int NEW_RELEASE = 1;
        public static final int REGULAR = 0;
        private int priceCode;

        private String title;
        public Movie(int priceCode, String title) {
                this.priceCode = priceCode;
                this.title = title;
        }
        public int getPriceCode() {
                return priceCode;
        }
        public String getTitle() {
                return title;
        }
        public void setPriceCode(int priceCode) {
                this.priceCode = priceCode;
        }

}
```

20. Source > Format을 누른다. 코드가 다음과 같이 변형된다.

```
package com.videorent;

public class Movie {

        public static final int CHILDRENS = 2;
        public static final int NEW_RELEASE = 1;
        public static final int REGULAR = 0;
        private int priceCode;

        private String title;

        public Movie(int priceCode, String title) {
                this.priceCode = priceCode;
                this.title = title;
        }

        public int getPriceCode() {
                return priceCode;
        }

        public String getTitle() {
                return title;
        }

        public void setPriceCode(int priceCode) {
                this.priceCode = priceCode;
        }

}
```

Rental 클래스 작성

1. 프로젝트를 선택하고 오른쪽 마우스 버튼을 눌러서 New > Class를 선택한다. 다음과 같은 대화상자가 나타난다. Name에 Rental를 넣고 Finish 버튼을 누른다.



Source folder: RefactoringPrj/src
Package: (default)
[ ] Enclosing type:
Name: Rental
Modifiers: [● public] ○ default ○ private ○ protected
[ ] abstract [ ] final [ ] static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
[ ] public static void main(String[] args)
[ ] Constructors from superclass
[✓] Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
[ ] Generate comments

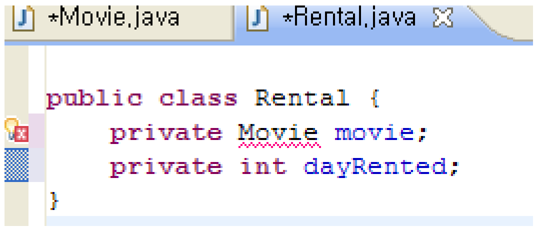
2. 다음과 같은 프로그램이 나타난다.

```java
public class Rental {

}
```

3. 다음과 같이 속성을 입력한다.

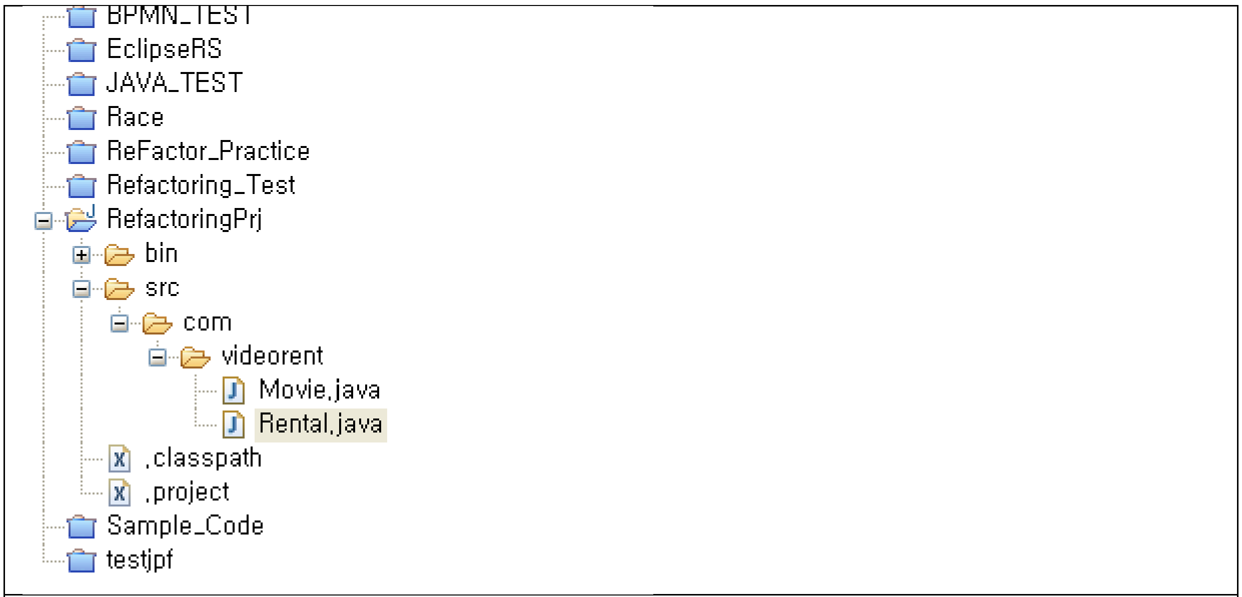
```java
public class Rental {
        private Movie movie;
        private int dayRented;
}
```

4. Movie 클래스의 패키지 위치가 다르기 때문에 다음 그림과 같이 에러가 표시된다.

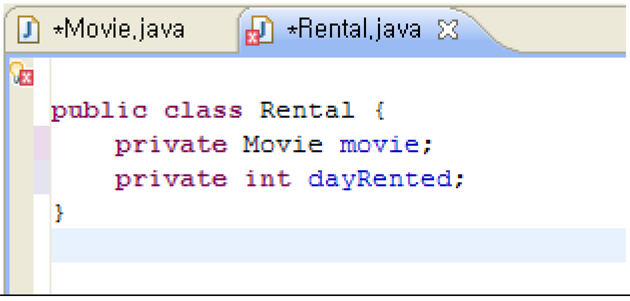


```java
public class Rental {
    private Movie movie;
    private int dayRented;
}
```

5. Rental 클래스의 패키지를 Movie 클래스의 패키지와 동일하게 만들어야 한다.
Navigator View에서 Rental 클래스를 드래그하여 videorent 폴더 아래 넣는다.

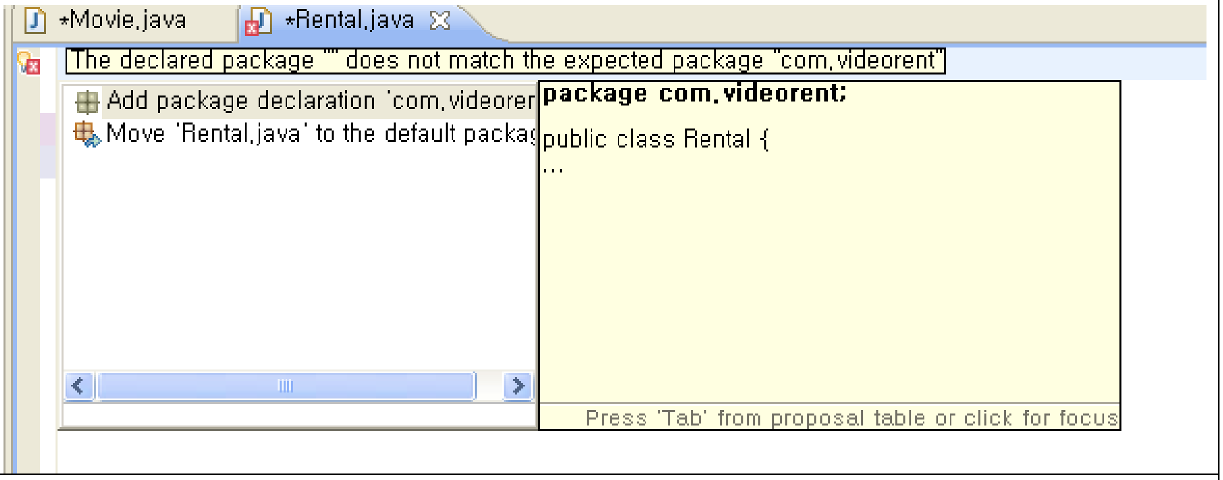


6. Rental 클래스에 다음과 같은 오류가 나타난다.



```
public class Rental {
    private Movie movie;
    private int dayRented;
}
```

7. 오류 표시를 눌러서 quickFix를 수행한다.


*Movie.java
*Rental.java
Add package declaration 'com.videorer
Move 'Rental.java' to the default packag
Press 'Tab' from proposal table or click for focus

The declared package "" does not match the expected package "com.videorent"

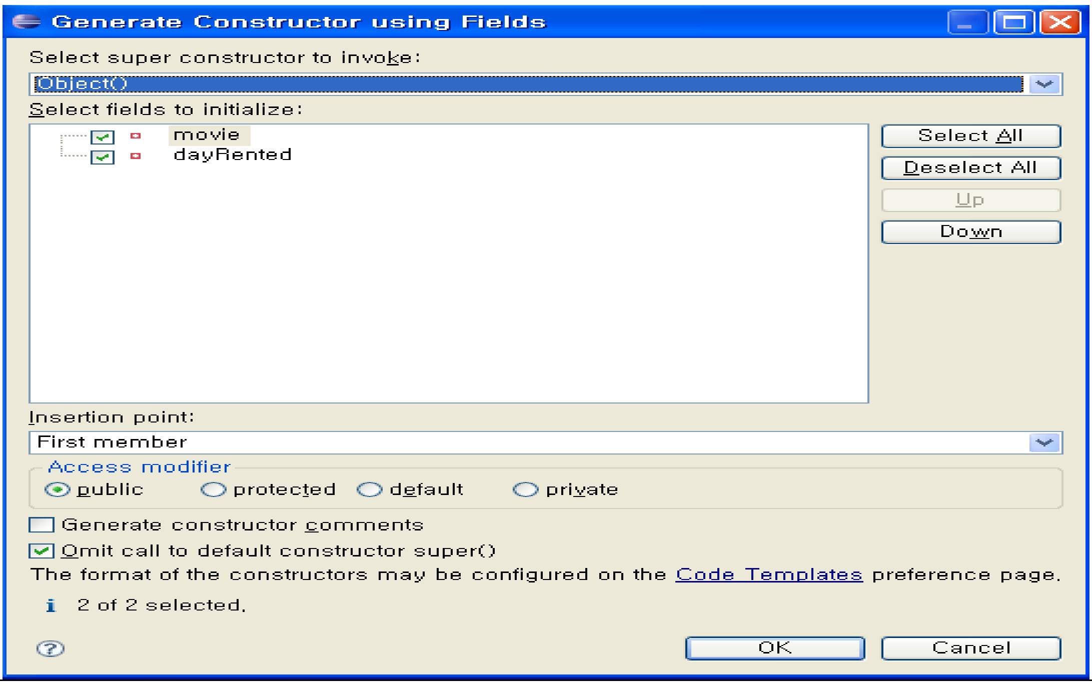
```
package com.videorent;
public class Rental {
...
```

8. Add package를 선택한다. 클래스가 다음과 같이 바뀐다.

```
package com.videorent;

public class Rental {
        private Movie movie;
        private int dayRented;
}
```

9. 편집기에서 Rental 클래스를 선택하고 오른쪽 마우스 버튼을 누르고 Source > Generate

Constructor using Fields를 선택한다.


Generate Constructor using Fields
Select super constructor to invoke:
Object()
Select fields to initialize:
movie
dayRented
Select All
Deselect All
Up
Down
Insertion point:
First member
Access modifier
public
protected
default
private
Generate constructor comments
Omit call to default constructor super()
The format of the constructors may be configured on the Code Templates preference page.
2 of 2 selected.
OK
Cancel

10. Omit call to default constructor super()를 선택하고 OK를 누른다.

11. Rental 클래스가 다음과 같이 변형된다.

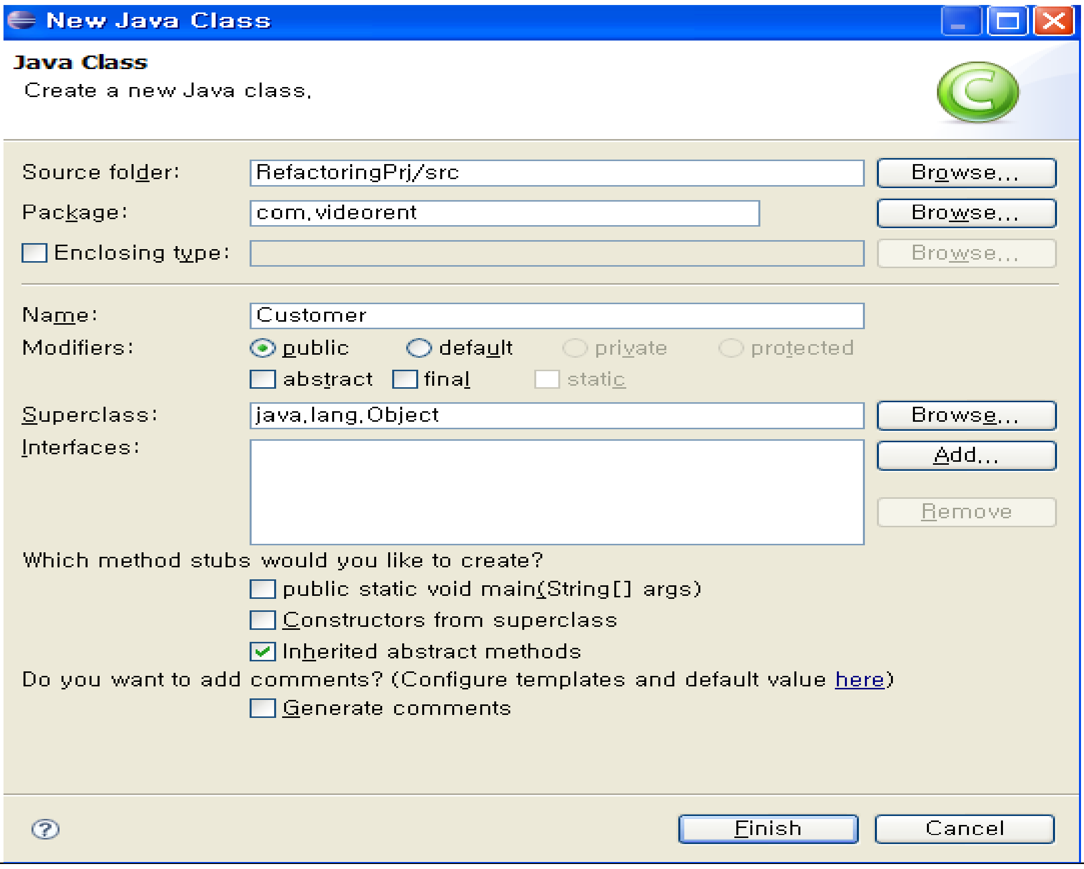
```
public class Rental {
	public Rental(Movie movie, int dayRented) {
		this.movie = movie;
		this.dayRented = dayRented;
	}
	private Movie movie;
	private int dayRented;
}
```

12. dayRented 필드에 get 메소드를 만든다. dayRented 필드를 선택하고 오른쪽 마우스 버튼을 눌러서 Source > Generate Getters and Setters를 선택한다. setDayRented는 check를 해제한다. Movie도 같은 방법으로 get 메소드를 만든다.

```
package com.videorent;

public class Rental {
	public Rental(Movie movie, int dayRented) {
		this.movie = movie;
		this.dayRented = dayRented;
	}
	private Movie movie;
	private int dayRented;
	public Movie getMovie() {
		return movie;
	}
	public int getDayRented() {
		return dayRented;
	}
}
```

Customer 클래스 작성

1. 프로젝트를 선택하고 오른쪽 마우스를 버튼을 누른 후 New > Class를 선택한다.
Package 명에는 com.videorent, Name에는 Customer를 넣는다.



Source folder: RefactoringPrj/src
Package: com.videorent
☐ Enclosing type:
Name: Customer
Modifiers: [✓ public] default private protected
☐ abstract ☐ final ☐ static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
☐ Generate comments

2. Finish 버튼을 누르면 다음과 같은 코드가 생성된다.

```
package com.videorent;

public class Customer {

}
```

3. 다음과 같이 속성을 추가한다.

```
package com.videorent;

public class Customer {
	private String name;
	private Vector<Rental> rentals = new Vector<Rental>();
}
```

4. 3번 코드는 Vector에 대한 import가 설정되어 있지 않다. Ctrl + Shift + O를 눌러서 import 문을 추가한다.

```
package com.videorent;

import java.util.Vector;

public class Customer {
	private String name;
	private Vector<Rental> rentals = new Vector<Rental>();
}
```

5. addRental 메소드를 다음과 같이 추가한다.

```
package com.videorent;

import java.util.Vector;

public class Customer {
	private String name;
	private Vector<Rental> rentals = new Vector<Rental>();

	public void addRental(Rental arg){
		rentals.addElement(arg);
	}
}
```

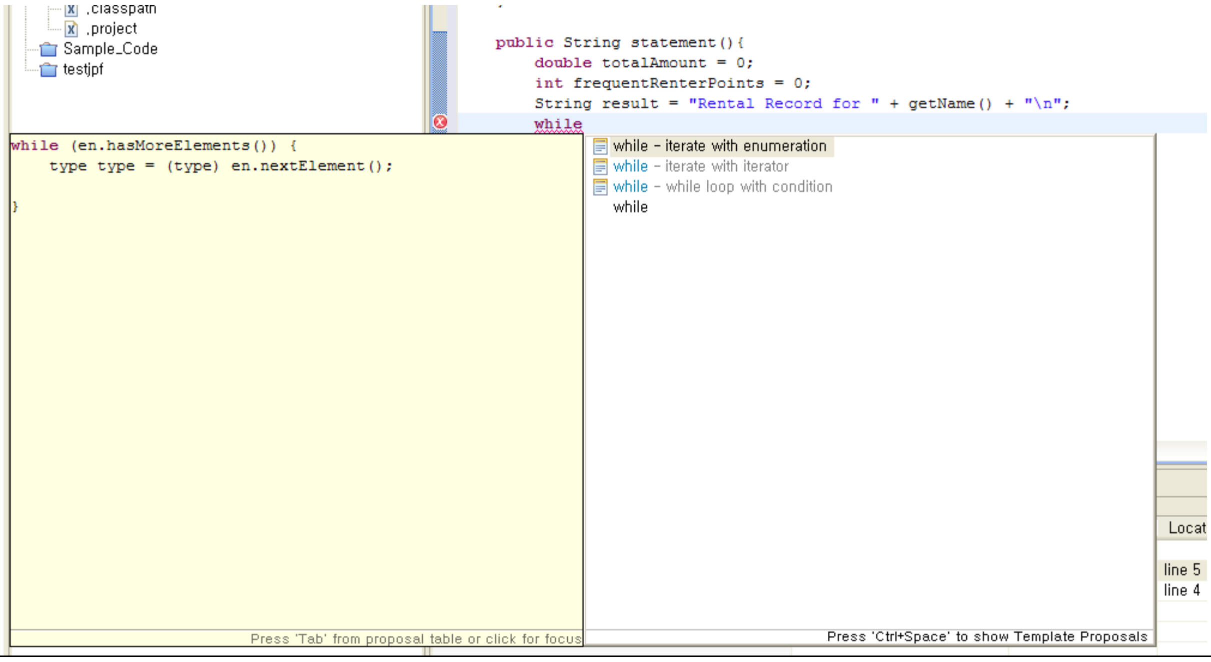
6. name의 get 메소드를 추가한다.

```
package com.videorent;

import java.util.Vector;

public class Customer {
	private String name;
	public String getName() {
		return name;
	}

	private Vector<Rental> rentals = new Vector<Rental>();

	public void addRental(Rental arg){
		rentals.addElement(arg);
	}
}
```

7. 다음과 같이 statement 메소드를 추가한다.

```
package com.videorent;

import java.util.Vector;

public class Customer {
	private String name;
	public String getName() {
		return name;
	}

	private Vector<Rental> rentals = new Vector<Rental>();

	public void addRental(Rental arg){
		rentals.addElement(arg);
	}

	public String statement(){
		double totalAmount = 0;
		int frequentRenterPoints = 0;
		String result = "Rental Record for " + getName() + "\n";
	}
}
```

8. while을 입력하고 Ctrl + Space를 누른다.



```
public String statement(){
    double totalAmount = 0;
    int frequentRenterPoints = 0;
    String result = "Rental Record for " + getName() + "\n";
    while
```

```
while (en.hasMoreElements()) {
    type type = (type) en.nextElement();

}
```

.classpath
.project
Sample_Code
testjpf
while - iterate with enumeration
while - iterate with iterator
while - while loop with condition
while
Press 'Tab' from proposal table or click for focus
Press 'Ctrl+Space' to show Template Proposals
Locat
line 5
line 4

9. while-iteration with enumeration을 선택하고 다음과 같이 입력한다.

```
package com.videorent;

import java.util.Enumeration;
import java.util.Vector;

public class Customer {
	private String name;
	public String getName() {
		return name;
	}

	private Vector<Rental> rentals = new Vector<Rental>();

	public void addRental(Rental arg){
		rentals.addElement(arg);
	}

	public String statement(){
		double totalAmount = 0;
		int frequentRenterPoints = 0;
		String result = "Rental Record for " + getName() + "\n";
		Enumeration<Rental> rentalss = rentals.elements();
		while (rentalss.hasMoreElements()) {
			double thisAMount = 0;
			Rental each = (Rental)rentalss.nextElement();

		}
	}
}
```

10. while 문 아래 switch를 입력하고 Ctrl + Space로 switch 문을 완성한다.

```
package com.videorent;

import java.util.Enumeration;
import java.util.Vector;

public class Customer {
	private String name;
	public String getName() {
		return name;
	}

	private Vector<Rental> rentals = new Vector<Rental>();

	public void addRental(Rental arg){
		rentals.addElement(arg);
	}

	public String statement(){
		double totalAmount = 0;
		int frequentRenterPoints = 0;
		String result = "Rental Record for " + getName() + "\n";
		Enumeration<Rental> rentalss = rentals.elements();
		while (rentalss.hasMoreElements()) {
			double thisAMount = 0;
			Rental each = (Rental)rentalss.nextElement();
			switch (key) {
			case value:

				break;

			default:
				break;
			}

		}
	}
}
```

11. Switch 내부를 다음과 같이 작성한다.

```java
package com.videorent;

import java.util.Enumeration;
import java.util.Vector;

public class Customer {
	private String name;
	public String getName() {
		return name;
	}

	private Vector<Rental> rentals = new Vector<Rental>();

	public void addRental(Rental arg){
		rentals.addElement(arg);
	}

	public String statement() {
		double totalAmount = 0;
		int frequentRenterPoints = 0;
		String result = "Rental Record for " + getName() + "\n";
		Enumeration<Rental> rentalss = rentals.elements();
		while (rentalss.hasMoreElements()) {
			double thisAmount = 0;
			Rental each = (Rental) rentalss.nextElement();
			switch (each.getMovie().getPriceCode()) {
				case Movie.REGULAR:
					thisAmount += 2;
					if (each.getDayRented() > 2)
						thisAmount += (each.getDayRented() - 2) * 1.5;
					break;
				case Movie.NEW_RELEASE:
					thisAmont += each.getDaysRented() * 3;
					break;
				case Movie.CHILDRENS:
					thisAmount += 1.5;
					if (each.getDayRented() > 3)
						thisAmount += (each.getDayRented() - 3) * 1.5;
			}

		}
	}
}
```

12. switch 문 다음에 포인트 적립에 관련된 로직을 추가한다.

```java
package com.videorent;

import java.util.Enumeration;
import java.util.Vector;

public class Customer {
	private String name;
	public String getName() {
		return name;
	}

	private Vector<Rental> rentals = new Vector<Rental>();

	public void addRental(Rental arg){
		rentals.addElement(arg);
	}

	public String statement() {
		double totalAmount = 0;
		int frequentRenterPoints = 0;
		String result = "Rental Record for " + getName() + "\n";
		Enumeration<Rental> rentalss = rentals.elements();
		while (rentalss.hasMoreElements()) {
			double thisAmount = 0;
			Rental each = (Rental) rentalss.nextElement();
			switch (each.getMovie().getPriceCode()) {
				case Movie.REGULAR:
					thisAmount += 2;
					if (each.getDayRented() > 2)
						thisAmount += (each.getDayRented() - 2) * 1.5;
					break;
				case Movie.NEW_RELEASE:
					thisAmont += each.getDaysRented() * 3;
					break;
				case Movie.CHILDRENS:
					thisAmount += 1.5;
					if (each.getDayRented() > 3)
						thisAmount += (each.getDayRented() - 3) * 1.5;
			}
			frequentRenterPoints ++;
			if ( (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
					each.getDayRented() > 1) frequentRenterPoints ++;

			result += "\t" + each.getMovie().getTitle() + "\t" +
				String.valueOf(thisAmount) + "\n";
			totalAmount += thisAmount;

		}


	}
}
```

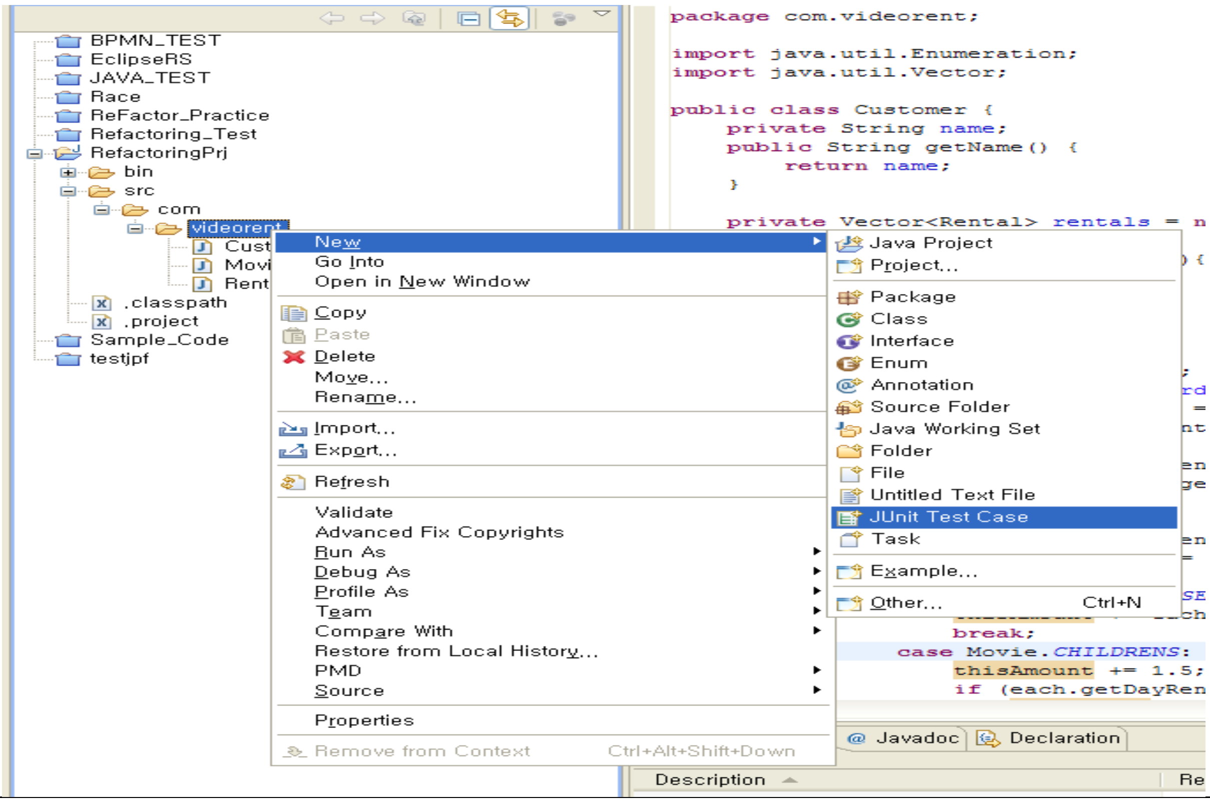
13. while 문 다음에 다음 로직을 추가한다.

```
package com.videorent;

import java.util.Enumeration;
import java.util.Vector;

public class Customer {
	private String name;
	public String getName() {
		return name;
	}
```

```
        private Vector<Rental> rentals = new Vector<Rental>();

        public void addRental(Rental arg){
                rentals.addElement(arg);
        }

        public String statement() {
                double totalAmount = 0;
                int frequentRenterPoints = 0;
                String result = "Rental Record for " + getName() + "\n";
                Enumeration<Rental> rentalss = rentals.elements();
                while (rentalss.hasMoreElements()) {
                        double thisAmount = 0;
                        Rental each = (Rental) rentalss.nextElement();
                        switch (each.getMovie().getPriceCode()) {
                                case Movie.REGULAR:
                                        thisAmount += 2;
                                        if (each.getDayRented() > 2)
                                                thisAmount += (each.getDayRented() - 2) * 1.5;
                                        break;
                                case Movie.NEW_RELEASE:
                                        thisAmont += each.getDaysRented() * 3;
                                        break;
                                case Movie.CHILDRENS:
                                        thisAmount += 1.5;
                                        if (each.getDayRented() > 3)
                                                thisAmount += (each.getDayRented() - 3) * 1.5;
                        }
                        frequentRenterPoints ++;
                        if ( (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
                                        each.getDayRented() > 1) frequentRenterPoints ++;

                        result += "\t" + each.getMovie().getTitle() + "\t" +
                                String.valueOf(thisAmount) + "\n";
                        totalAmount += thisAmount;

                }

                result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
                result += "You earned" + String.valueOf(frequentRenterPoints) +
                        " frequent renter points";
                return result;


        }
}
```

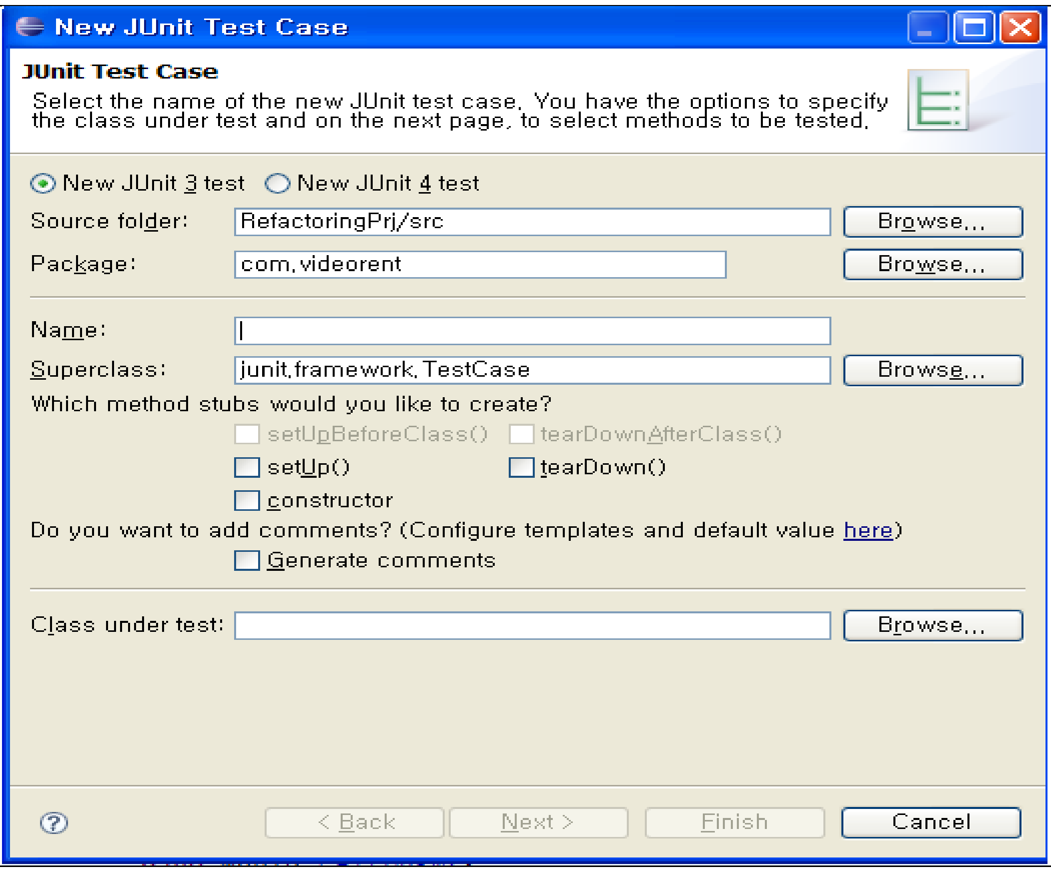
단위 테스트 수행

1. videorent 폴더를 선택한 후 New > JUnitTestCase를 선택한다.



```
package com.videorent;

import java.util.Enumeration;
import java.util.Vector;

public class Customer {
    private String name;
    public String getName() {
        return name;
    }

    private Vector<Rental> rentals = n
```

2. 다음과 같은 대화상자가 나타난다.


New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. You have the options to specify the class under test and on the next page, to select methods to be tested.

[✓ New JUnit 3 test] New JUnit 4 test

Source folder: RefactoringPrj/src

Package: com.videorent

Name:

Superclass: junit.framework.TestCase

Which method stubs would you like to create?

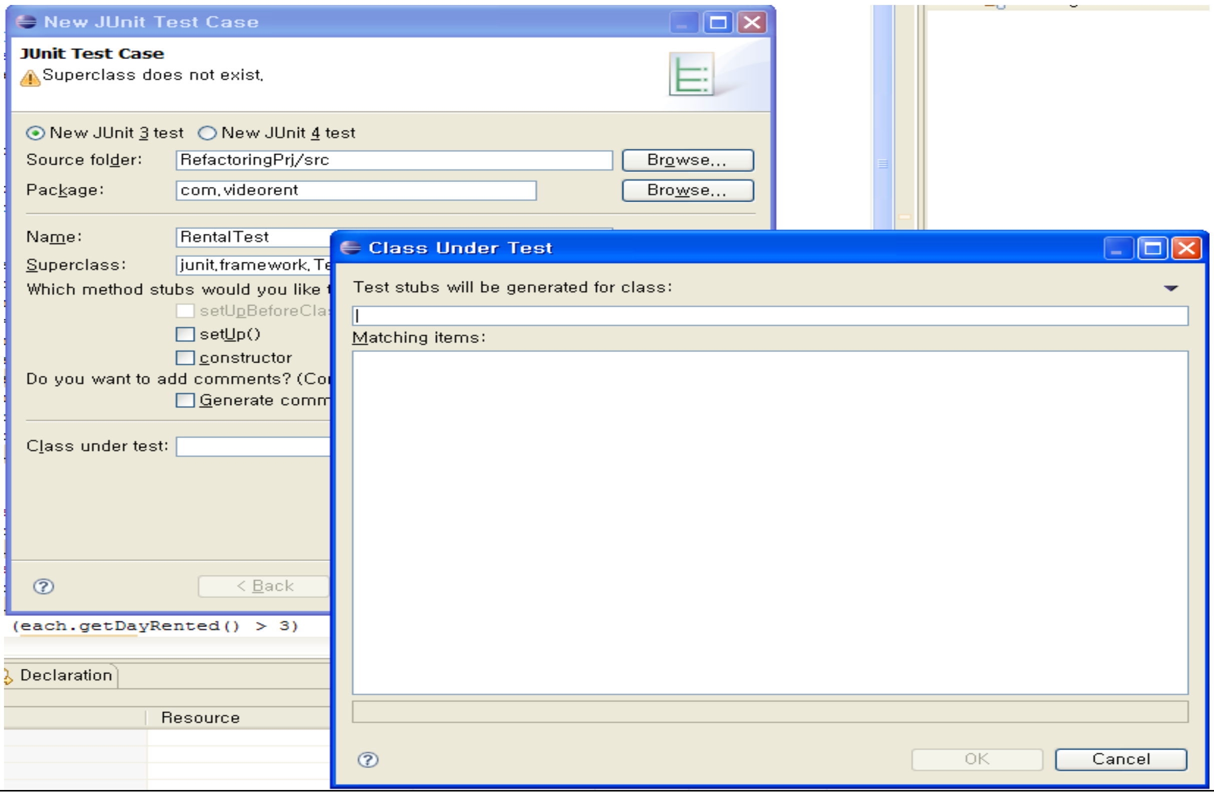
- [ ] setUpBeforeClass()
- [ ] tearDownAfterClass()
- [ ] setUp()
- [ ] tearDown()
- [ ] constructor

Do you want to add comments? (Configure templates and default value here)

- [ ] Generate comments

Class under test:

3. Name에 RentalTest를 입력하고 Class under test의 browse 버튼을 누른다.


New JUnit Test Case

JUnit Test Case

Superclass does not exist.

[✓ New JUnit 3 test] New JUnit 4 test

Source folder: RefactoringPrj/src

Package: com.videorent

Name: RentalTest

Superclass: junit.framework.Te

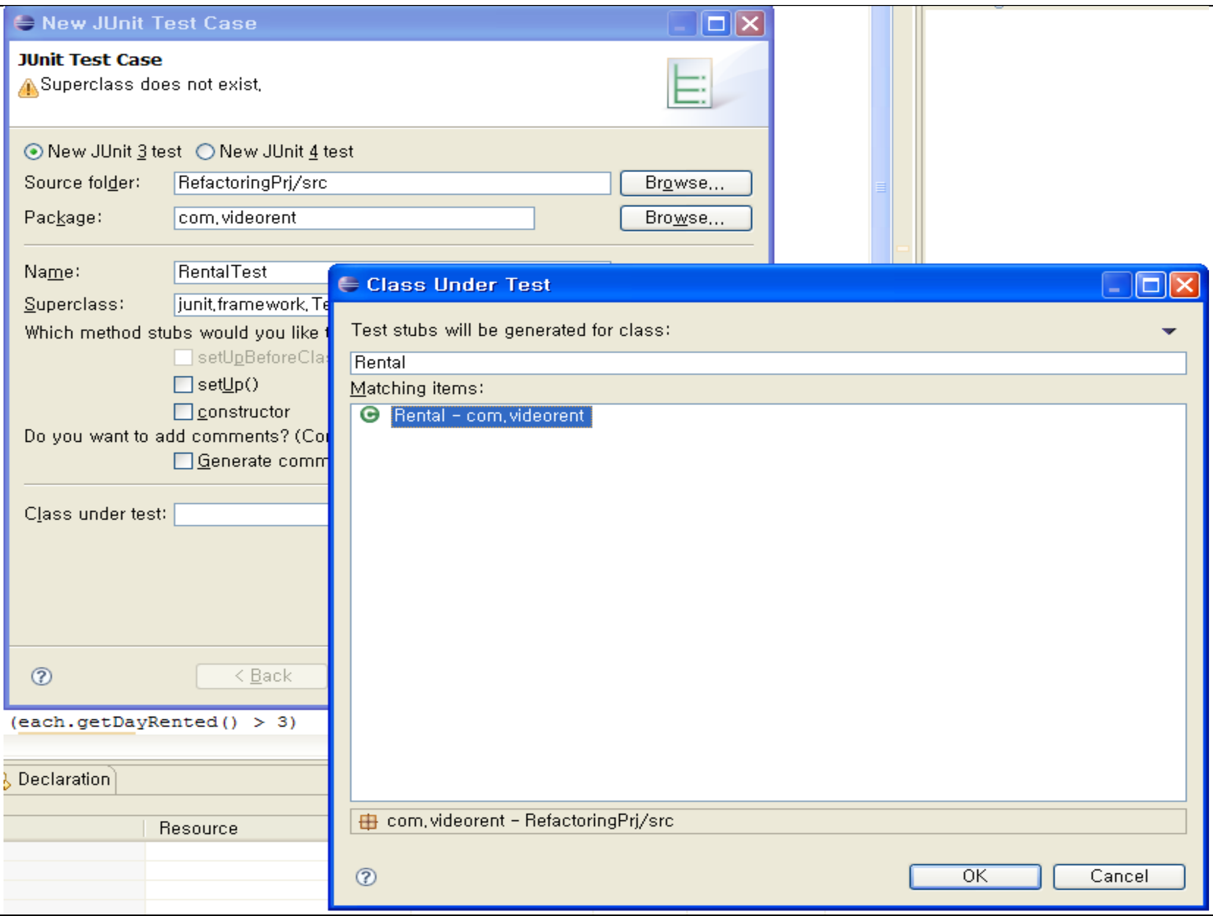
Class Under Test

Test stubs will be generated for class:

Matching items:

```
(each.getDayRented() > 3)
```

4. Test할 클래스 이름에 Rental을 넣으면 Matching Items에 클래스가 나타난다. 클래스를 선택하고 OK를 누른다.


New JUnit Test Case

JUnit Test Case

Superclass does not exist.

(●) New JUnit 3 test ( ) New JUnit 4 test

Source folder: RefactoringPrj/src

Package: com.videorent

Name: RentalTest

Superclass: junit.framework.Te

Which method stubs would you like t

[ ] setUpBeforeCla

[ ] setUp()

[ ] constructor

Do you want to add comments? (Co

[ ] Generate comm

Class under test:

```
(each.getDayRented() > 3)
```

Class Under Test

Test stubs will be generated for class:

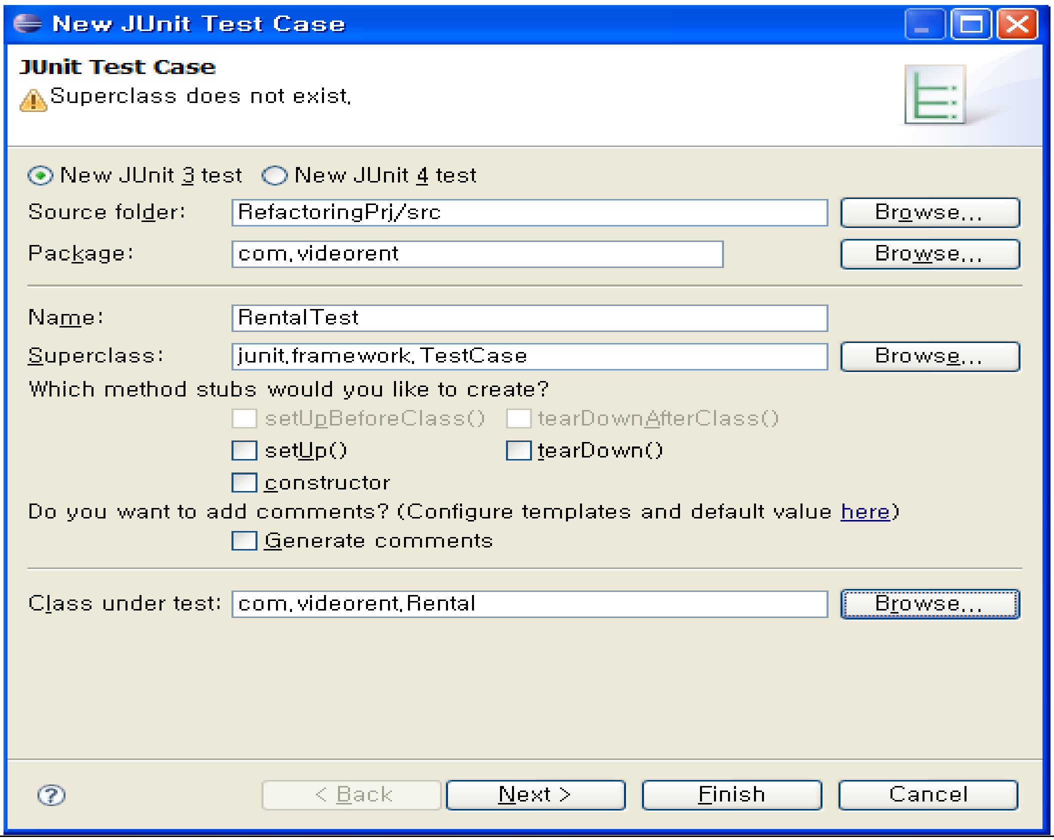
Rental

Matching items:

Rental - com.videorent

com.videorent - RefactoringPrj/src

5. 대화상자가 다음과 같이 보인다.


New JUnit Test Case

JUnit Test Case

Superclass does not exist.

(●) New JUnit 3 test ( ) New JUnit 4 test

Source folder: RefactoringPrj/src

Package: com.videorent

Name: RentalTest

Superclass: junit.framework.TestCase

Which method stubs would you like to create?

[ ] setUpBeforeClass() [ ] tearDownAfterClass()

[ ] setUp() [ ] tearDown()

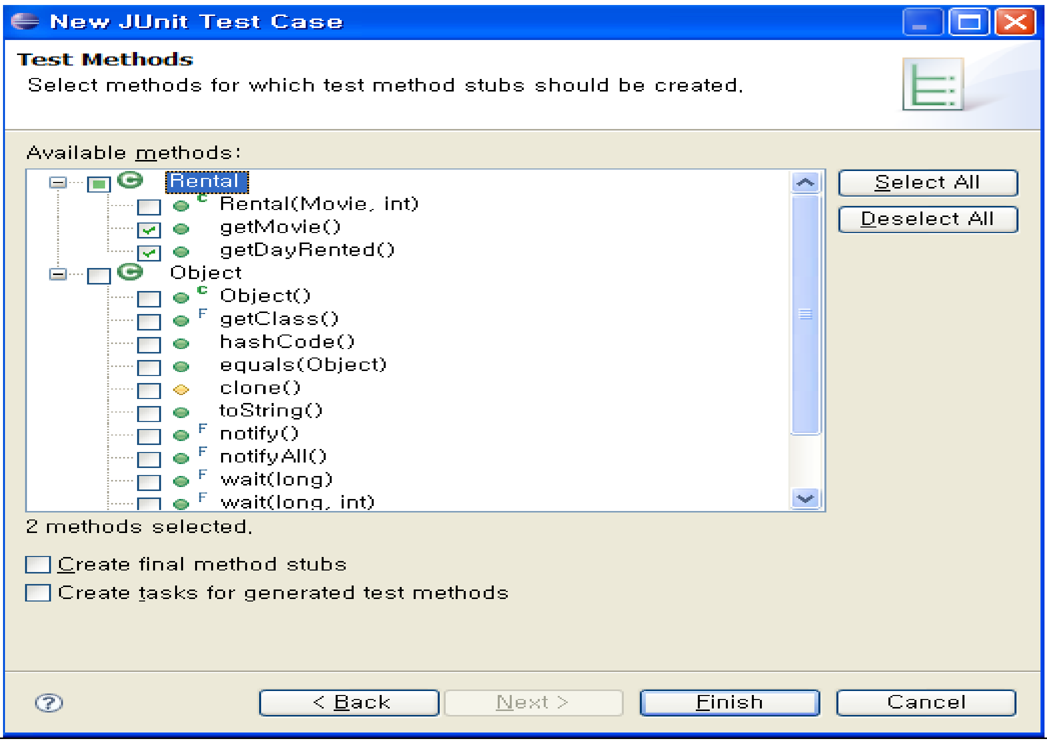
[ ] constructor

Do you want to add comments? (Configure templates and default value here)

[ ] Generate comments

Class under test: com.videorent.Rental

6. Next 버튼을 누른다. 테스트할 메소드를 선택하는 창이 나타난다. Rental 클래스의 getMovie와 getDayRented를 선택한다.



New JUnit Test Case

Test Methods

Select methods for which test method stubs should be created.

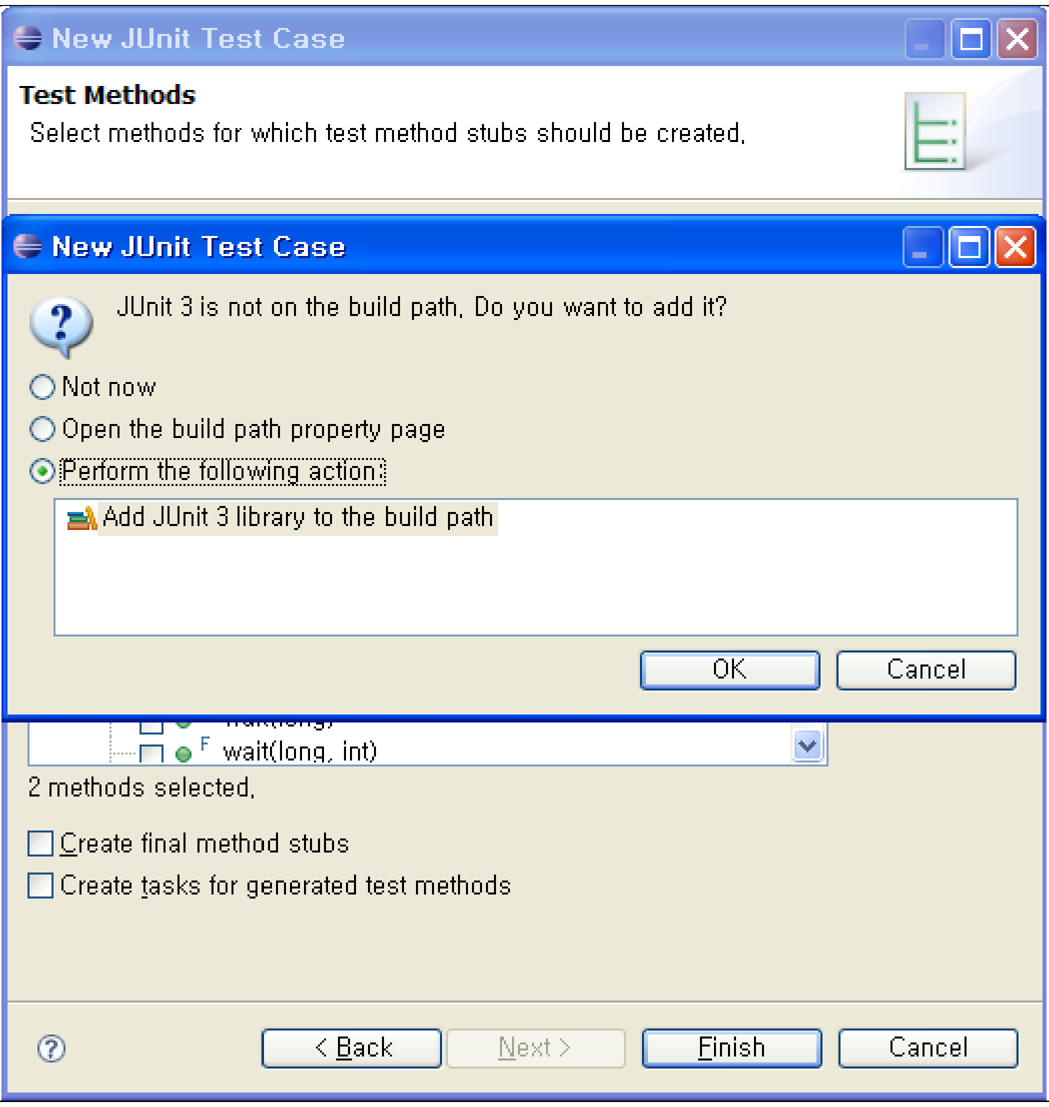
Available methods:

- Rental
  - [ ] Rental(Movie, int)
  - [✓] getMovie()
  - [✓] getDayRented()
- [ ] Object
  - [ ] Object()
  - [ ] getClass()
  - [ ] hashCode()
  - [ ] equals(Object)
  - [ ] clone()
  - [ ] toString()
  - [ ] notify()
  - [ ] notifyAll()
  - [ ] wait(long)
  - [ ] wait(long, int)

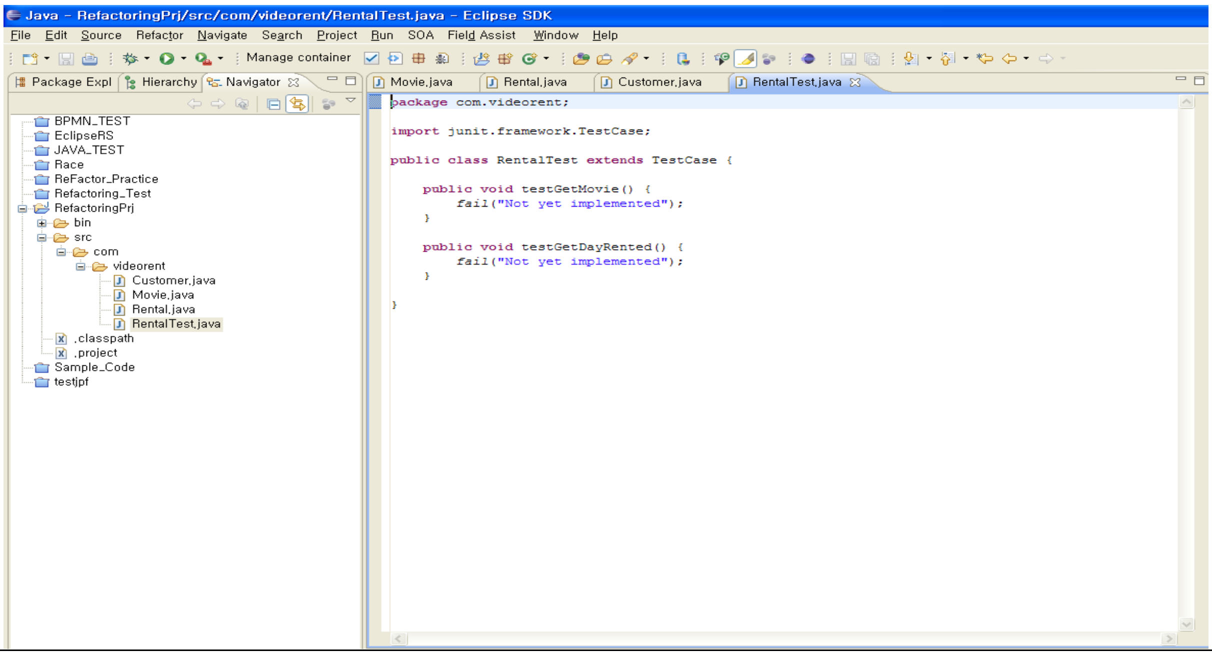
2 methods selected.

- [ ] Create final method stubs
- [ ] Create tasks for generated test methods

7. 완료 버튼을 누른다. 다음 창이 나오면 OK를 누른다.


New JUnit Test Case
Test Methods
Select methods for which test method stubs should be created.
New JUnit Test Case
JUnit 3 is not on the build path. Do you want to add it?
Not now
Open the build path property page
Perform the following action:
Add JUnit 3 library to the build path
OK
Cancel
wait(long, int)
2 methods selected.
Create final method stubs
Create tasks for generated test methods
< Back
Next >
Finish
Cancel

8. 다음과 같은 소스가 작성된다.

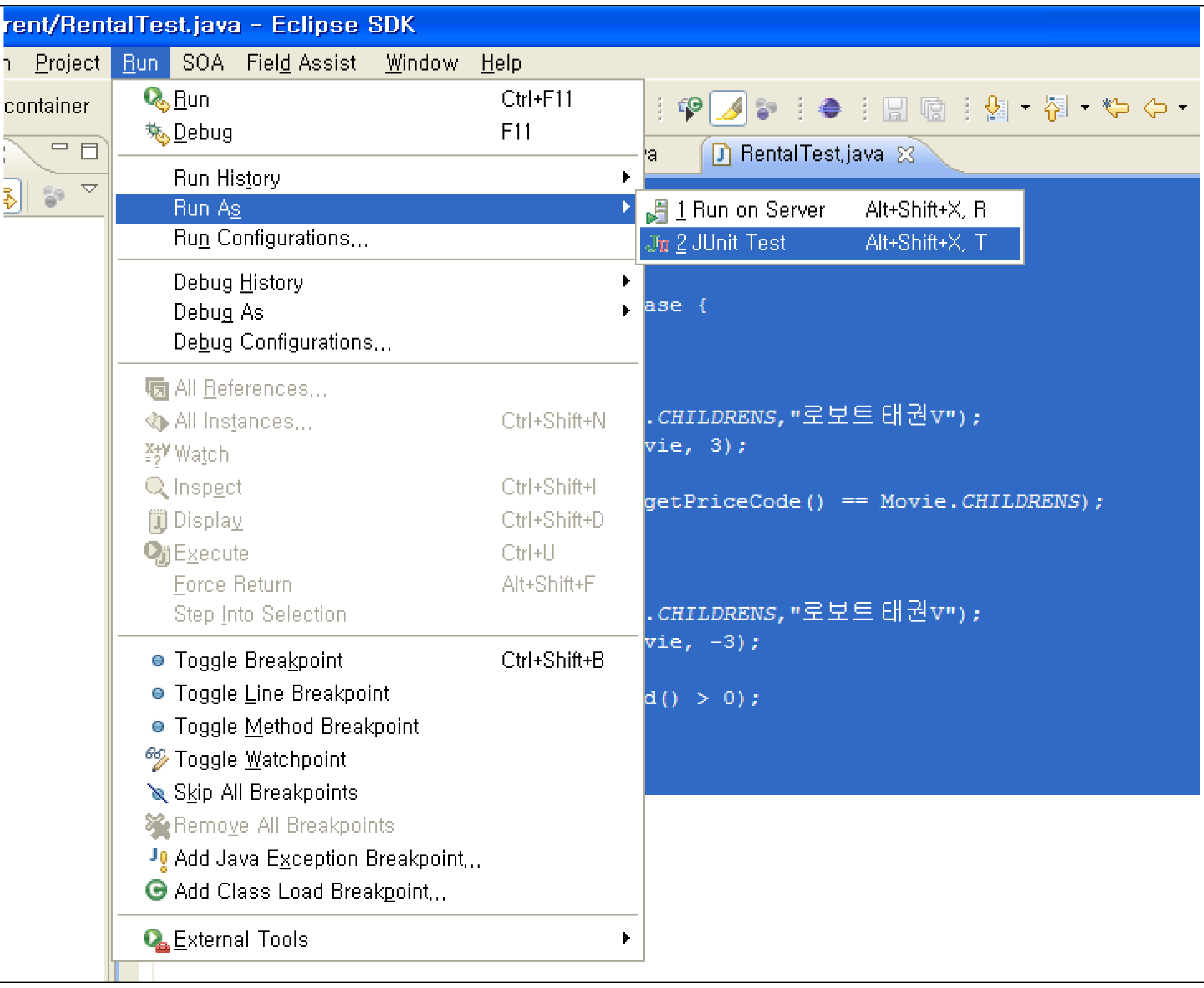


```
package com.videorent;

import junit.framework.TestCase;

public class RentalTest extends TestCase {

    public void testGetMovie() {
        fail("Not yet implemented");
    }

    public void testGetDayRented() {
        fail("Not yet implemented");
    }

}
```

9. 소스를 다음과 같이 수정한다.

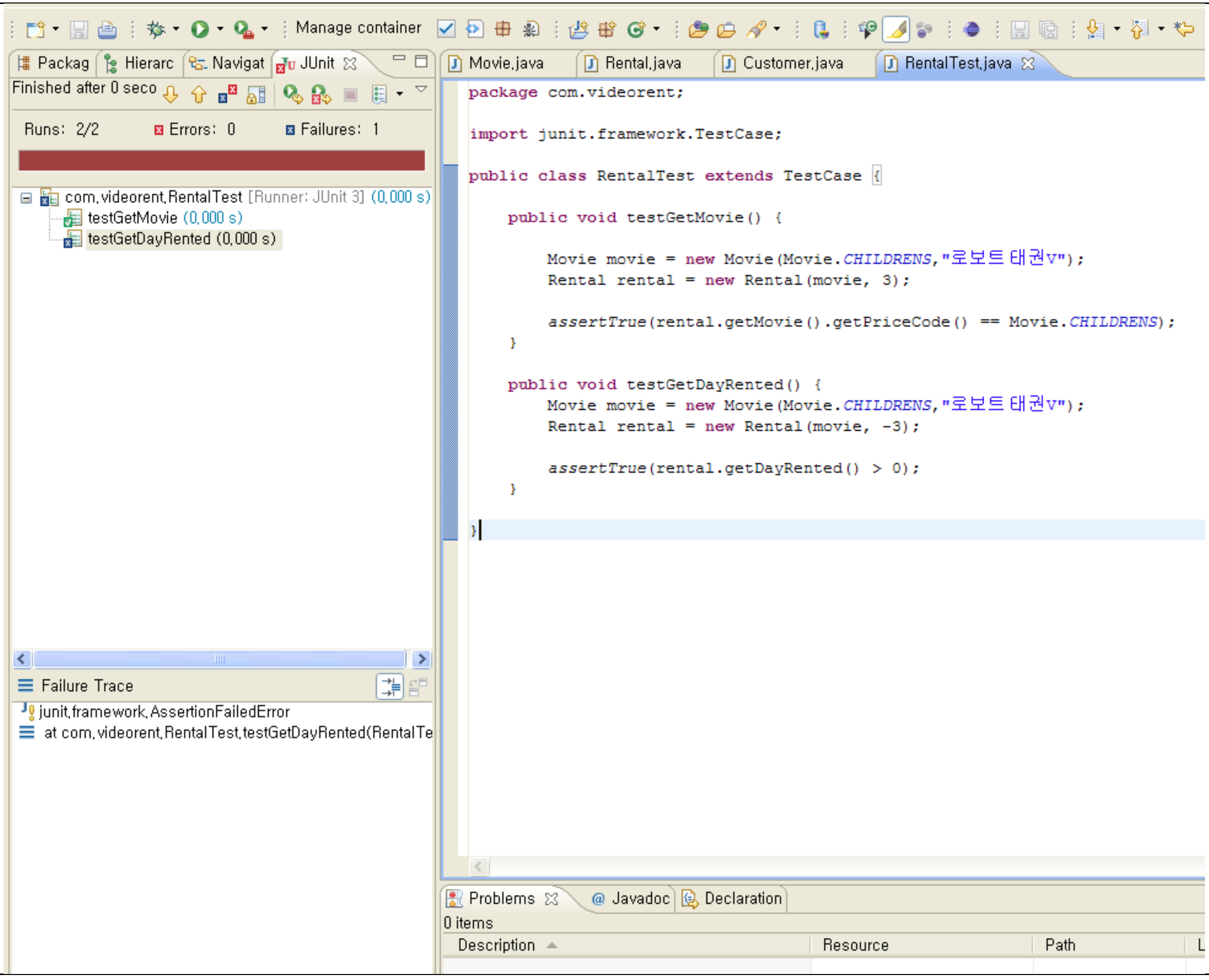
```java
package com.videorent;

import junit.framework.TestCase;

public class RentalTest extends TestCase {

	public void testGetMovie() {

		Movie movie = new Movie(Movie.CHILDRENS,"로보트 태권V");
		Rental rental = new Rental(movie, 3);

		assertTrue(rental.getMovie().getPriceCode() == Movie.CHILDRENS);
	}

	public void testGetDayRented() {
		Movie movie = new Movie(Movie.CHILDRENS,"로보트 태권V");
		Rental rental = new Rental(movie, -3);

		assertTrue(rental.getDayRented() > 0);
	}

}
```

10. Run 메뉴에서 RunAs > JUnitTest를 실행한다.



```
ase {

.CHILDRENS,"로보트 태권V");
vie, 3);

getPriceCode() == Movie.CHILDRENS);

.CHILDRENS,"로보트 태권V");
vie, -3);

d() > 0);
```

11. 테스트 결과는 다음과 같다.



```
Finished after 0 seco
Runs: 2/2    Errors: 0    Failures: 1
com.videorent.RentalTest [Runner: JUnit 3] (0.000 s)
    testGetMovie (0.000 s)
    testGetDayRented (0.000 s)

Failure Trace
junit.framework.AssertionFailedError
at com.videorent.RentalTest.testGetDayRented(RentalTe
```

```java
package com.videorent;

import junit.framework.TestCase;

public class RentalTest extends TestCase {

    public void testGetMovie() {

        Movie movie = new Movie(Movie.CHILDRENS,"로보트 태권v");
        Rental rental = new Rental(movie, 3);

        assertTrue(rental.getMovie().getPriceCode() == Movie.CHILDRENS);
    }

    public void testGetDayRented() {
        Movie movie = new Movie(Movie.CHILDRENS,"로보트 태권v");
        Rental rental = new Rental(movie, -3);

        assertTrue(rental.getDayRented() > 0);
    }

}
```

```
0 items
Description    Resource    Path    L
```

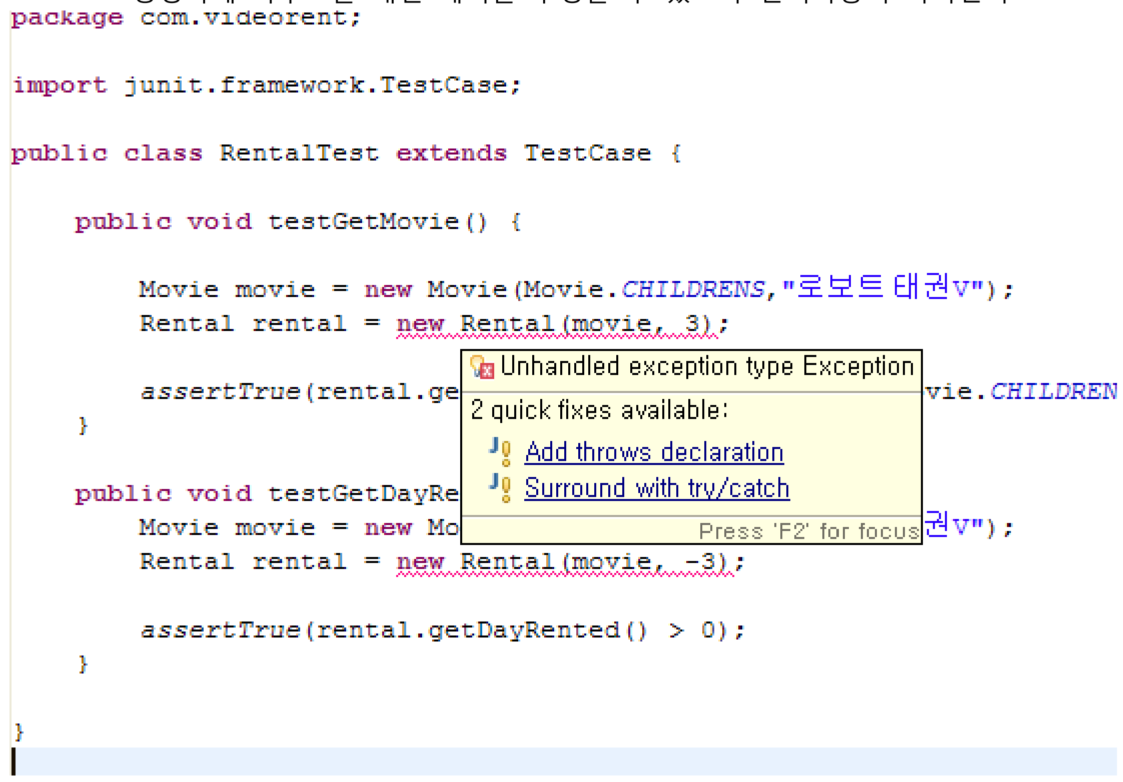
12. 대출 일자에 0과 음수가 들어가지 않도록 방지해야 한다. 대출 일자에 0과 음수가 들어가면 exception을 던지도록 다음과 같이 Rental 클래스를 수정한다.

```java
package com.videorent;

public class Rental {

	public Rental(Movie movie, int dayRented) throws Exception {

		if(dayRented <= 0 ){
			throw new Exception("대출일자에 음수가 들어갔습니다.");
		}

		this.movie = movie;
		this.dayRented = dayRented;
	}
	private Movie movie;
	private int dayRented;
	public Movie getMovie() {
		return movie;
	}
	public int getDayRented() {
		return dayRented;
	}
}
```

13. RentalTest 클래스는 다음과 같이 생성자에 에러가 발생한다.

```
package com.videorent;

import junit.framework.TestCase;

public class RentalTest extends TestCase {

	public void testGetMovie() {

		Movie movie = new Movie(Movie.CHILDRENS,"로보트 태권V");
		Rental rental = new Rental(movie, 3);

		assertTrue(rental.getMovie().getPriceCode() == Movie.CHILDRENS);
	}

	public void testGetDayRented() {
		Movie movie = new Movie(Movie.CHILDRENS,"로보트 태권V");
		Rental rental = new Rental(movie, -3);

		assertTrue(rental.getDayRented() > 0);
	}

}
```

14. Rental 생성자에 마우스를 대면 에러를 수정할 수 있도록 선택사항이 나타난다.



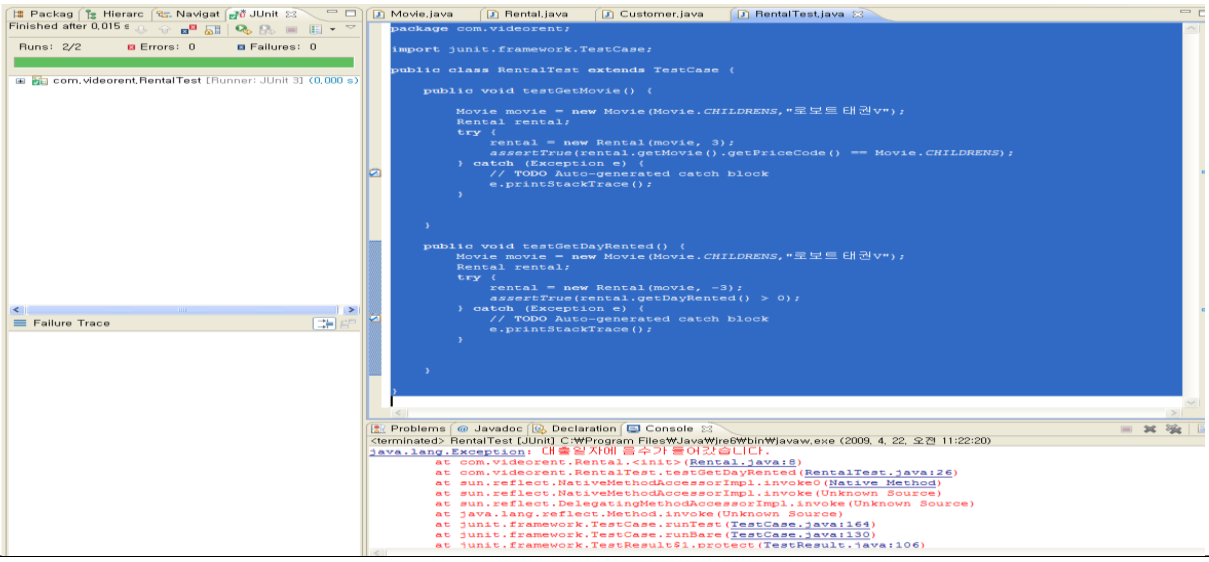
```
package com.videorent;

import junit.framework.TestCase;

public class RentalTest extends TestCase {

    public void testGetMovie() {

        Movie movie = new Movie(Movie.CHILDRENS,"로보트 태권V");
        Rental rental = new Rental(movie, 3);

        assertTrue(rental.ge                                  vie.CHILDREN
    }

    public void testGetDayRe
        Movie movie = new Mo                                  권V");
        Rental rental = new Rental(movie, -3);

        assertTrue(rental.getDayRented() > 0);
    }

}
```

Unhandled exception type Exception
2 quick fixes available:
Add throws declaration
Surround with try/catch
Press 'F2' for focus

15. Surround with try/catch를 선택하면 코드가 다음과 같이 바뀐다. assert 문은 try 문 안에 넣는다.

```java
package com.videorent;

import junit.framework.TestCase;

public class RentalTest extends TestCase {

	public void testGetMovie() {

		Movie movie = new Movie(Movie.CHILDRENS,"로보트 태권V");
		Rental rental;
		try {
			rental = new Rental(movie, 3);
			assertTrue(rental.getMovie().getPriceCode() == Movie.CHILDRENS);
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}


	}

	public void testGetDayRented() {
		Movie movie = new Movie(Movie.CHILDRENS,"로보트 태권V");
		Rental rental;
		try {
			rental = new Rental(movie, -3);
			assertTrue(rental.getDayRented() > 0);
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}


	}
}
```

16. . Run 메뉴에서 RunAs > JUnitTest를 실행한다.



```
Finished after 0,015 s
Runs: 2/2    Errors: 0    Failures: 0
com.videorent.RentalTest [Runner: JUnit 3] (0,000 s)
```

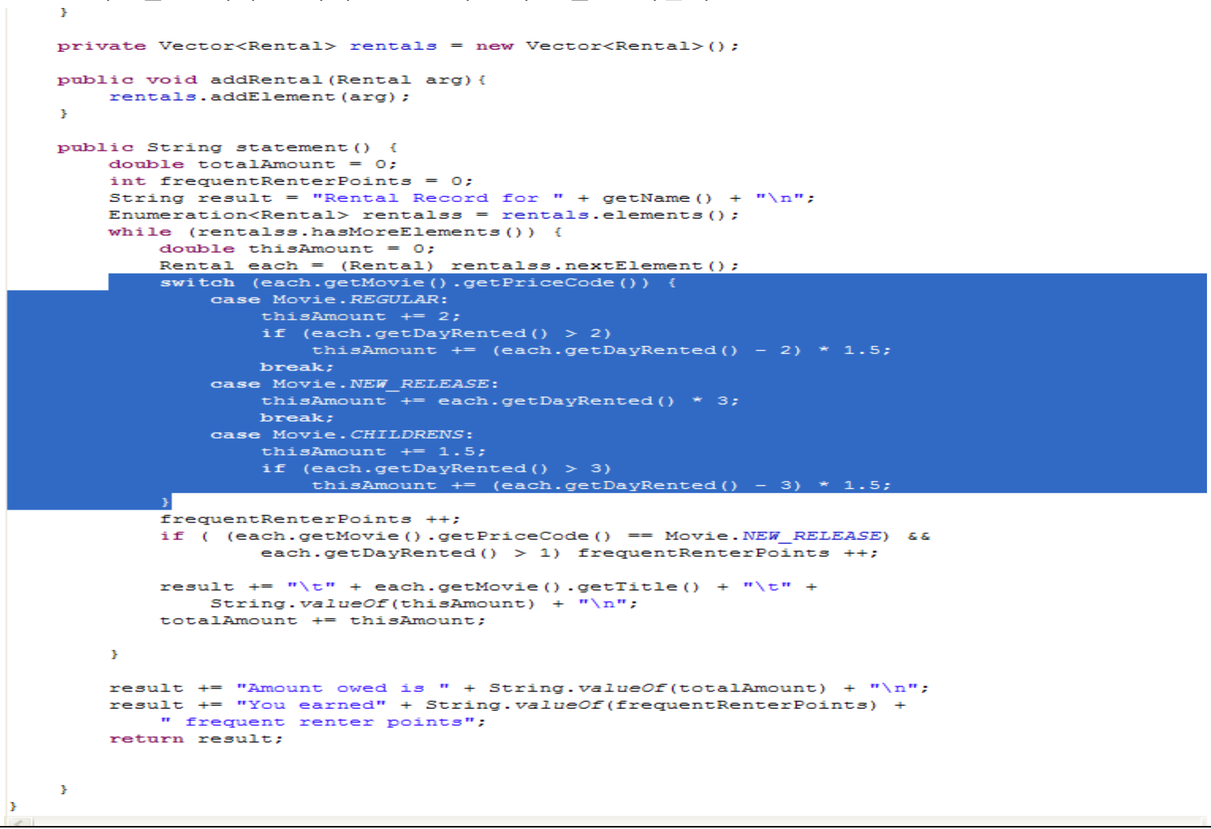
```
<terminated> RentalTest [JUnit] C:\Program Files\Java\jre6\bin\javaw.exe (2009, 4, 22, 오전 11:22:20)
java.lang.Exception: 대출일자에 음수가 들어갔습니다.
	at com.videorent.Rental.<init>(Rental.java:8)
	at com.videorent.RentalTest.testGetDayRented(RentalTest.java:26)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
	at java.lang.reflect.Method.invoke(Unknown Source)
	at junit.framework.TestCase.runTest(TestCase.java:164)
	at junit.framework.TestCase.runBare(TestCase.java:130)
	at junit.framework.TestResult$1.protect(TestResult.java:106)
```

Refactoring 일단계

1. Customer의 statement 메소드를 refactoring 한다. 회색으로 표시된 부분을 별도의 메소드로 추출한다.

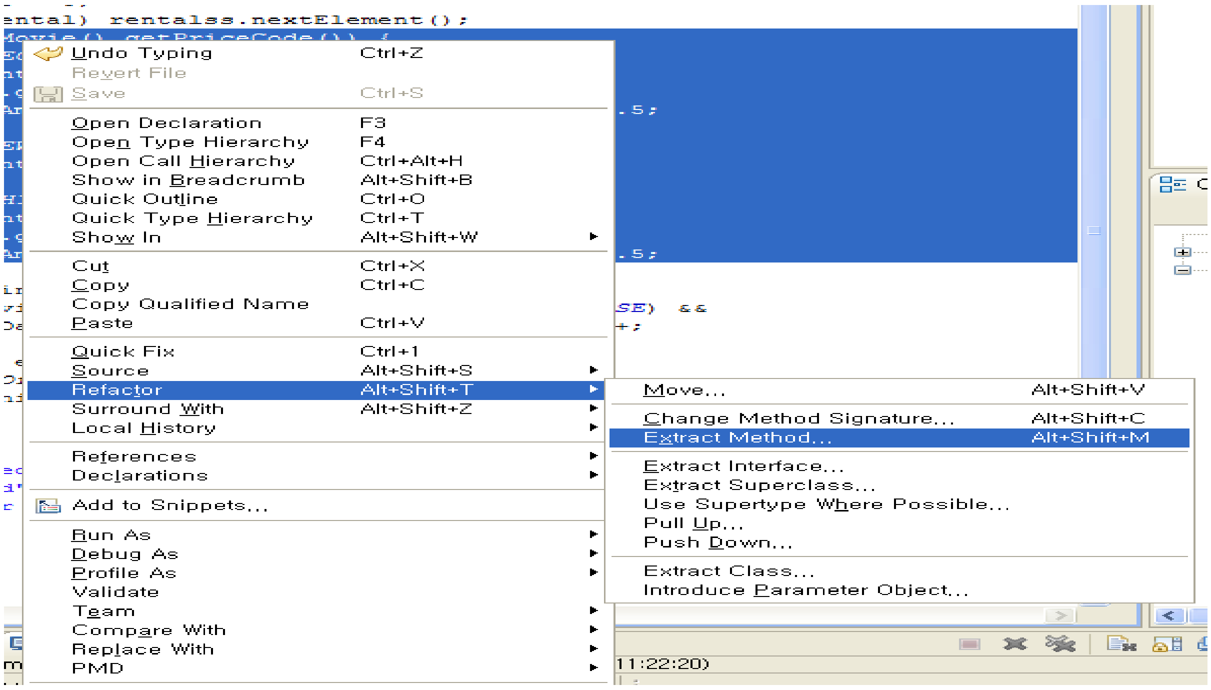
```
    public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
        String result = "Rental Record for " + getName() + "\n";
        Enumeration<Rental> rentalss = rentals.elements();
        while (rentalss.hasMoreElements()) {
            double thisAmount = 0;
            Rental each = (Rental) rentalss.nextElement();
            switch (each.getMovie().getPriceCode()) {
                case Movie.REGULAR:
                    thisAmount += 2;
                    if (each.getDayRented() > 2)
                        thisAmount += (each.getDayRented() - 2) * 1.5;
                    break;
                case Movie.NEW_RELEASE:
                    thisAmount += each.getDayRented() * 3;
                    break;
                case Movie.CHILDRENS:
                    thisAmount += 1.5;
                    if (each.getDayRented() > 3)
                        thisAmount += (each.getDayRented() - 3) * 1.5;
            }
            frequentRenterPoints ++;
            if ( (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
                    each.getDayRented() > 1) frequentRenterPoints ++;

            result += "\t" + each.getMovie().getTitle() + "\t" +
                String.valueOf(thisAmount) + "\n";
            totalAmount += thisAmount;

        }

        result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned" + String.valueOf(frequentRenterPoints) +
            " frequent renter points";
        return result;


    }
```

2. Customer 클래스를 선택하고 회색으로 표시된 부분을 선택한다.

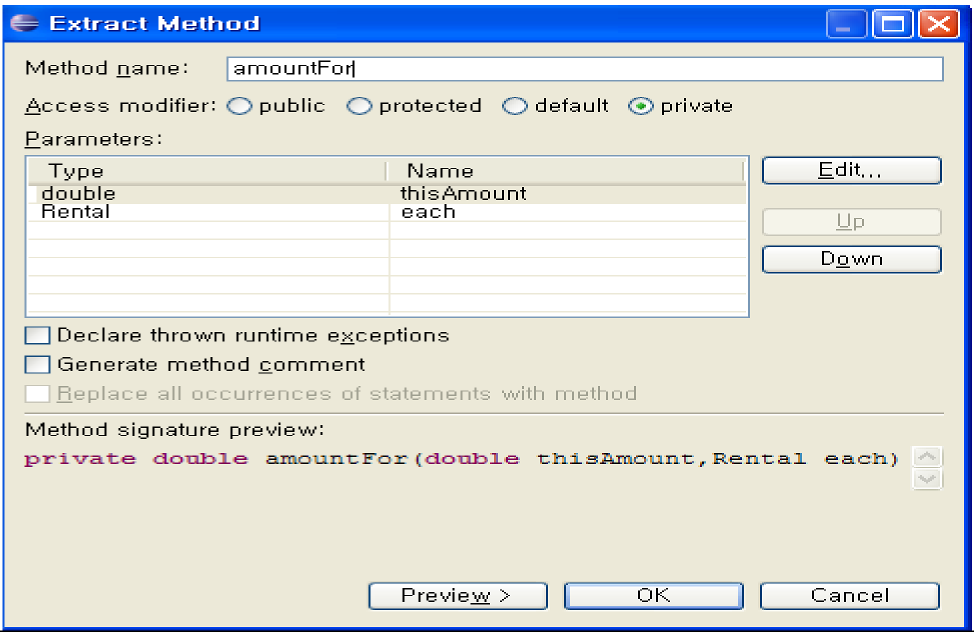


```
    }

    private Vector<Rental> rentals = new Vector<Rental>();

    public void addRental(Rental arg){
        rentals.addElement(arg);
    }

    public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
        String result = "Rental Record for " + getName() + "\n";
        Enumeration<Rental> rentalss = rentals.elements();
        while (rentalss.hasMoreElements()) {
            double thisAmount = 0;
            Rental each = (Rental) rentalss.nextElement();
            switch (each.getMovie().getPriceCode()) {
                case Movie.REGULAR:
                    thisAmount += 2;
                    if (each.getDayRented() > 2)
                        thisAmount += (each.getDayRented() - 2) * 1.5;
                    break;
                case Movie.NEW_RELEASE:
                    thisAmount += each.getDayRented() * 3;
                    break;
                case Movie.CHILDRENS:
                    thisAmount += 1.5;
                    if (each.getDayRented() > 3)
                        thisAmount += (each.getDayRented() - 3) * 1.5;
            }
            frequentRenterPoints ++;
            if ( (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
                    each.getDayRented() > 1) frequentRenterPoints ++;

            result += "\t" + each.getMovie().getTitle() + "\t" +
                String.valueOf(thisAmount) + "\n";
            totalAmount += thisAmount;

        }

        result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
        result += "You earned" + String.valueOf(frequentRenterPoints) +
            " frequent renter points";
        return result;


    }
}
```

3. 코드를 선택한 상태에서 마우스 오른쪽 버튼을 클릭하여 Refactor > Extract Method를 선택한다.



4. 대화상자가 나타나면 name에 amountFor를 입력한다.



Method name: amountFor

Access modifier: ( ) public ( ) protected ( ) default [✓ private]

Parameters:

Type	Name
double	thisAmount
Rental	each

☐ Declare thrown runtime exceptions
☐ Generate method comment
☐ Replace all occurrences of statements with method

Method signature preview:

```
private double amountFor(double thisAmount,Rental each)
```

5. OK를 누르면 프로그램이 다음과같이 변형된다.

```
package com.videorent;

import java.util.Enumeration;
import java.util.Vector;

public class Customer {
	private String name;
	public String getName() {
		return name;
	}

	private Vector<Rental> rentals = new Vector<Rental>();

	public void addRental(Rental arg){
		rentals.addElement(arg);
	}

	public String statement() {
		double totalAmount = 0;
		int frequentRenterPoints = 0;
		String result = "Rental Record for " + getName() + "\n";
		Enumeration<Rental> rentalss = rentals.elements();
		while (rentalss.hasMoreElements()) {
			double thisAmount = 0;
			Rental each = (Rental) rentalss.nextElement();
			thisAmount = amountFor(thisAmount, each);
			frequentRenterPoints ++;
			if ( (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
					each.getDayRented() > 1) frequentRenterPoints ++;

			result += "\t" + each.getMovie().getTitle() + "\t" +
				String.valueOf(thisAmount) + "\n";
```

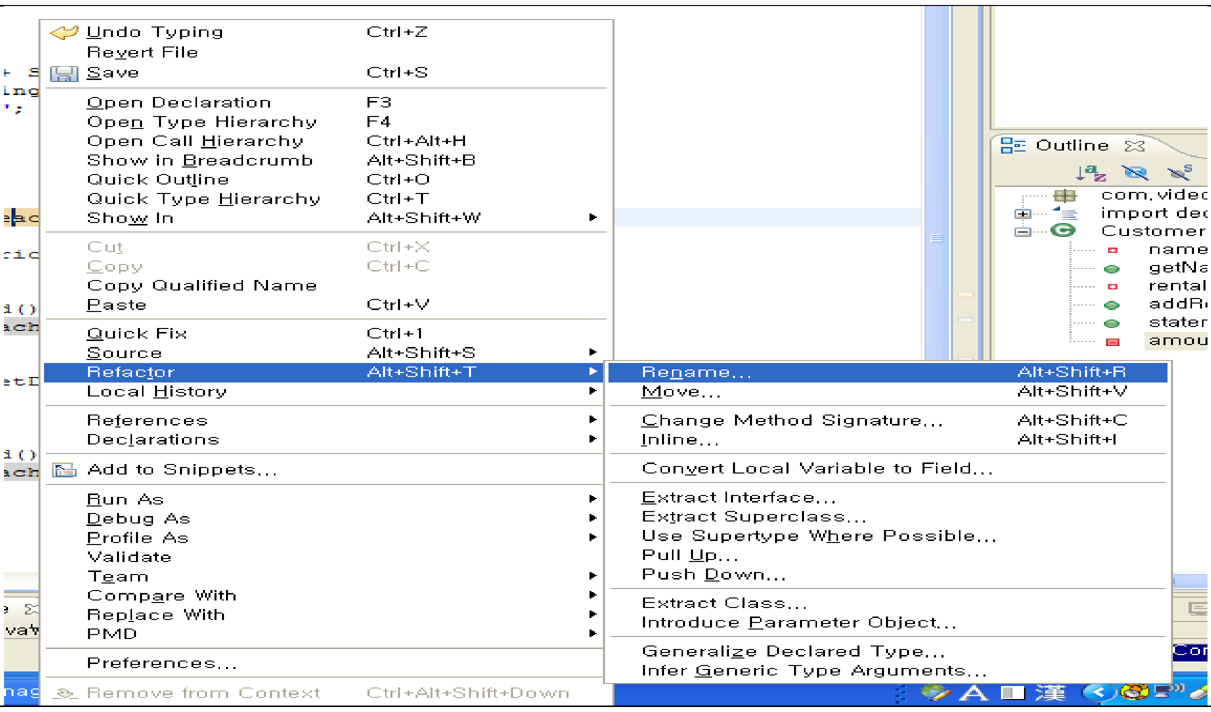
```
                    totalAmount += thisAmount;

             }

             result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
             result += "You earned" + String.valueOf(frequentRenterPoints) +
                    " frequent renter points";
             return result;


      }

      private double amountFor(double thisAmount, Rental each) {
             switch (each.getMovie().getPriceCode()) {
                    case Movie.REGULAR:
                           thisAmount += 2;
                           if (each.getDayRented() > 2)
                                  thisAmount += (each.getDayRented() - 2) * 1.5;
                           break;
                    case Movie.NEW_RELEASE:
                           thisAmount += each.getDayRented() * 3;
                           break;
                    case Movie.CHILDRENS:
                           thisAmount += 1.5;
                           if (each.getDayRented() > 3)
                                  thisAmount += (each.getDayRented() - 3) * 1.5;
             }
             return thisAmount;
      }
}
```

6. amountFor 메소드에서 parameter thisAmount는 제거한다. 프로그램을 다음과 같이 수정한다.

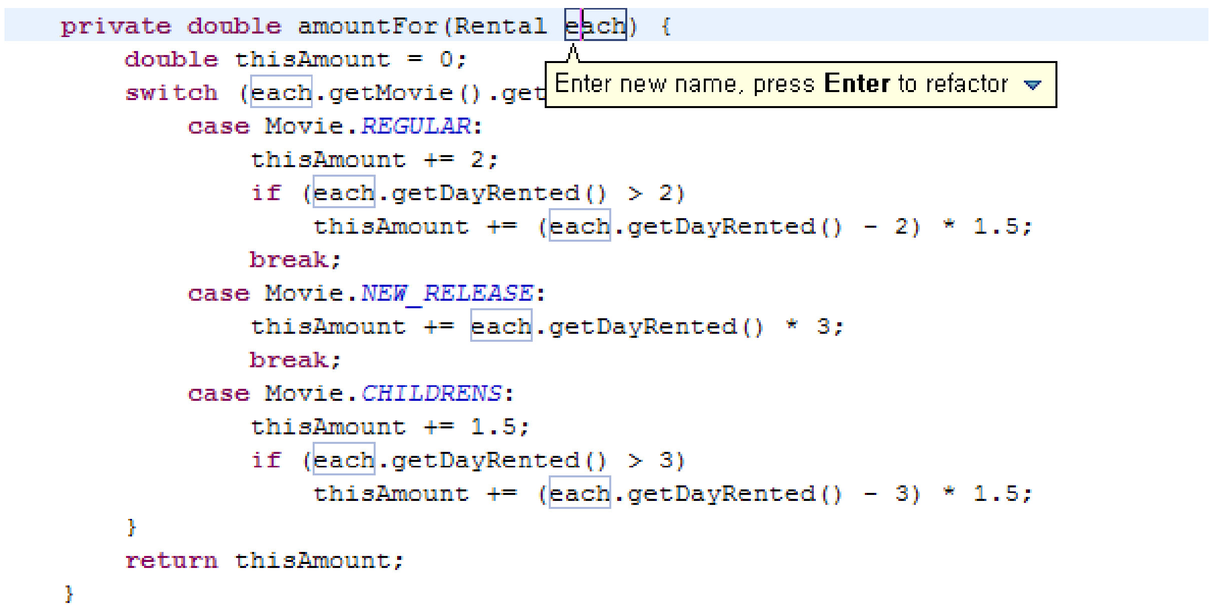
```
package com.videorent;

import java.util.Enumeration;
import java.util.Vector;

public class Customer {
      private String name;
      public String getName() {
             return name;
      }

      private Vector<Rental> rentals = new Vector<Rental>();

      public void addRental(Rental arg){
             rentals.addElement(arg);
      }

      public String statement() {
             double totalAmount = 0;
             int frequentRenterPoints = 0;
             String result = "Rental Record for " + getName() + "\n";
             Enumeration<Rental> rentalss = rentals.elements();
             while (rentalss.hasMoreElements()) {
                    double thisAmount = 0;
                    Rental each = (Rental) rentalss.nextElement();
                    thisAmount = amountFor(each);
                    frequentRenterPoints ++;
                    if ( (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
                                 each.getDayRented() > 1) frequentRenterPoints ++;

                    result += "\t" + each.getMovie().getTitle() + "\t" +
                           String.valueOf(thisAmount) + "\n";
                    totalAmount += thisAmount;

              }

              result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
              result += "You earned" + String.valueOf(frequentRenterPoints) +
                    " frequent renter points";
              return result;


       }

       private double amountFor(Rental each) {
              double thisAmount = 0;
              switch (each.getMovie().getPriceCode()) {
                    case Movie.REGULAR:
                           thisAmount += 2;
                           if (each.getDayRented() > 2)
                                  thisAmount += (each.getDayRented() - 2) * 1.5;
                           break;
                    case Movie.NEW_RELEASE:
                           thisAmount += each.getDayRented() * 3;
                           break;
                    case Movie.CHILDRENS:
                           thisAmount += 1.5;
                           if (each.getDayRented() > 3)
                                  thisAmount += (each.getDayRented() - 3) * 1.5;
              }
              return thisAmount;
       }
}
```

7. amountFor의 파라메터 명 each를 알아보기 쉽도록 aRental로 변형한다.

파라메터 each를 선택한 후 오른쪽 마우스 버튼을 누른다. Refactor > Rename을 선택한다.



8. 입력화면이 다음과 같이 바뀐다.



```
    private double amountFor(Rental each) {
        double thisAmount = 0;
        switch (each.getMovie().get
            case Movie.REGULAR:
                thisAmount += 2;
                if (each.getDayRented() > 2)
                    thisAmount += (each.getDayRented() - 2) * 1.5;
                break;
            case Movie.NEW_RELEASE:
                thisAmount += each.getDayRented() * 3;
                break;
            case Movie.CHILDRENS:
                thisAmount += 1.5;
                if (each.getDayRented() > 3)
                    thisAmount += (each.getDayRented() - 3) * 1.5;
        }
        return thisAmount;
    }
}
```

Enter new name, press **Enter** to refactor

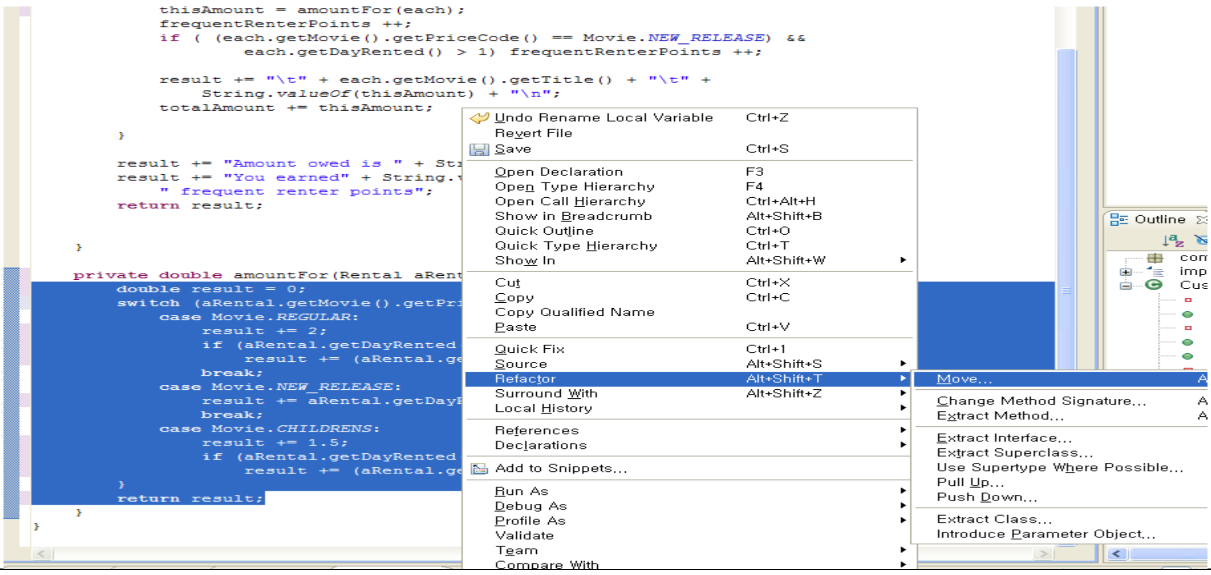
9. 이 상태에서 aRental을 입력하면 each가 들어간 부분이 모두 aRental로 바뀐다.

```
    private double amountFor(Rental aRental) {
        double thisAmount = 0;
        switch (aRental.getMovie().getPriceCode()) {
            case Movie.REGULAR:
                thisAmount += 2;
                if (aRental.getDayRented() > 2)
                    thisAmount += (aRental.getDayRented() - 2) * 1.5;
                break;
            case Movie.NEW_RELEASE:
                thisAmount += aRental.getDayRented() * 3;
                break;
            case Movie.CHILDRENS:
                thisAmount += 1.5;
                if (aRental.getDayRented() > 3)
                    thisAmount += (aRental.getDayRented() - 3) * 1.5;
        }
        return thisAmount;
    }
```

10. 같은 방법으로 지역 변수 thisAmount를 result로 바꾼다.

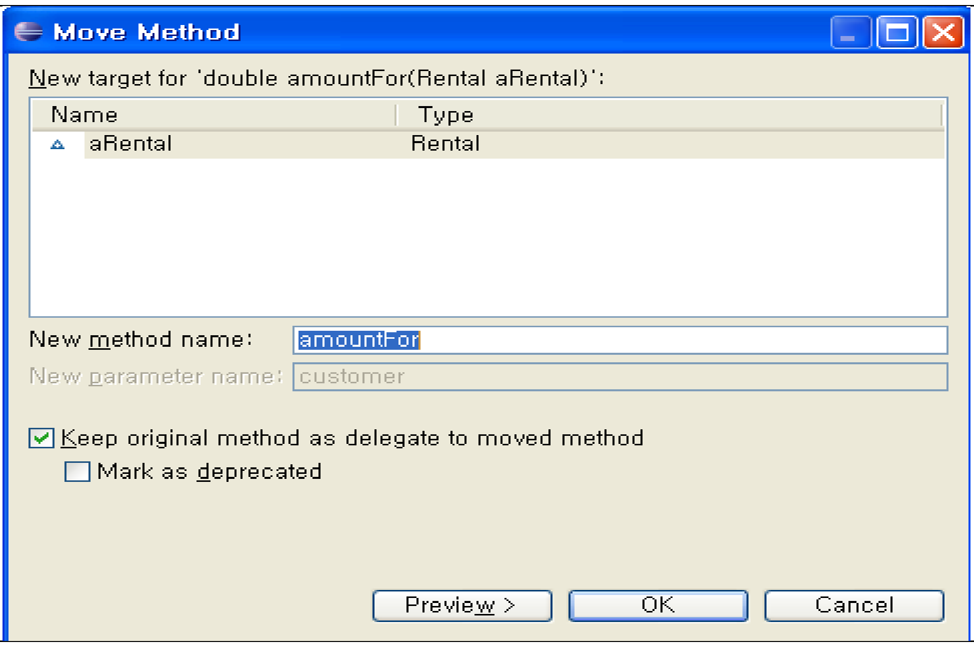
```
    private double amountFor(Rental aRental) {
            double result = 0;
            switch (aRental.getMovie().getPriceCode()) {
                case Movie.REGULAR:
                    result += 2;
                    if (aRental.getDayRented() > 2)
                        result += (aRental.getDayRented() - 2) * 1.5;
                    break;
                case Movie.NEW_RELEASE:
                    result += aRental.getDayRented() * 3;
                    break;
                case Movie.CHILDRENS:
                    result += 1.5;
                    if (aRental.getDayRented() > 3)
                        result += (aRental.getDayRented() - 3) * 1.5;
            }
            return result;
        }
```

11. Customer의 amountFor의 내용은 Rental 클래스에 들어가는 것이 바람직하다. 따라서 Customer의 amountFor는 남겨두고 내용은 Rental 클래스로 이동한다. amountFor 메소드의 내용을 선택하고 오른쪽 마우스 버튼을 누른다. Refactor > Move를 선택한다.



```
            thisAmount = amountFor(each);
            frequentRenterPoints ++;
            if ( (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
                    each.getDayRented() > 1) frequentRenterPoints ++;

            result += "\t" + each.getMovie().getTitle() + "\t" +
                String.valueOf(thisAmount) + "\n";
            totalAmount += thisAmount;

        }

        result += "Amount owed is " + St
        result += "You earned" + String.
            " frequent renter points";
        return result;

    }

    private double amountFor(Rental aRent
        double result = 0;
        switch (aRental.getMovie().getPri
            case Movie.REGULAR:
                result += 2;
                if (aRental.getDayRented
                    result += (aRental.ge
                break;
            case Movie.NEW_RELEASE:
                result += aRental.getDay
                break;
            case Movie.CHILDRENS:
                result += 1.5;
                if (aRental.getDayRented
                    result += (aRental.ge
        }
        return result;
    }
}
```

12. 다음과 같은 창이 나타난다.


Move Method

New target for 'double amountFor(Rental aRental)':

Name	Type
aRental	Rental

New method name: amountFor

New parameter name: customer

[✓] Keep original method as delegate to moved method

[ ] Mark as deprecated

13. Keep original method as delegate to moved method를 체크하고 OK를 누른다. method name은 getCharge로 바꾼다.

14. Customer 클래스는 다음과 같이 변형된다.

```
	private double amountFor(Rental aRental) {
		return aRental.getCharge();
	}
```

15. Rental 클래스는 다음과 같이 변형된다.

```
package com.videorent;

public class Rental {

	public Rental(Movie movie, int dayRented) throws Exception {

		if(dayRented <= 0 ){

			throw new Exception("대출일자에 음수가 들어갔습니다.");
		}

		this.movie = movie;
		this.dayRented = dayRented;
	}
	private Movie movie;
	private int dayRented;
	public Movie getMovie() {
		return movie;
	}
	public int getDayRented() {
		return dayRented;
	}
	double getCharge() {
		double result = 0;
		switch (getMovie().getPriceCode()) {
			case Movie.REGULAR:
```

```
                        result += 2;
                        if (getDayRented() > 2)
                                result += (getDayRented() - 2) * 1.5;
                        break;
                case Movie.NEW_RELEASE:
                        result += getDayRented() * 3;
                        break;
                case Movie.CHILDRENS:
                        result += 1.5;
                        if (getDayRented() > 3)
                                result += (getDayRented() - 3) * 1.5;
                }
                return result;
        }
}
```

16. Customer 클래스의 amountFor 메소드를 제거하고 each.getCharge()로 바꾼다.

```
        public String statement() {
                double totalAmount = 0;
                int frequentRenterPoints = 0;
                String result = "Rental Record for " + getName() + "\n";
                Enumeration<Rental> rentalss = rentals.elements();
                while (rentalss.hasMoreElements()) {
                        double thisAmount = 0;
                        Rental each = (Rental) rentalss.nextElement();
                        thisAmount = each.getCharge();
                        frequentRenterPoints ++;
                        if ( (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
                                each.getDayRented() > 1) frequentRenterPoints ++;

                        result += "\t" + each.getMovie().getTitle() + "\t" +
                                String.valueOf(thisAmount) + "\n";
                        totalAmount += thisAmount;

                }

                result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
                result += "You earned" + String.valueOf(frequentRenterPoints) +
                        " frequent renter points";
                return result;


        }
```

17. statement 메소드의 thisAmount도 제거하고 each.getCharge로 모두 바꾼다.

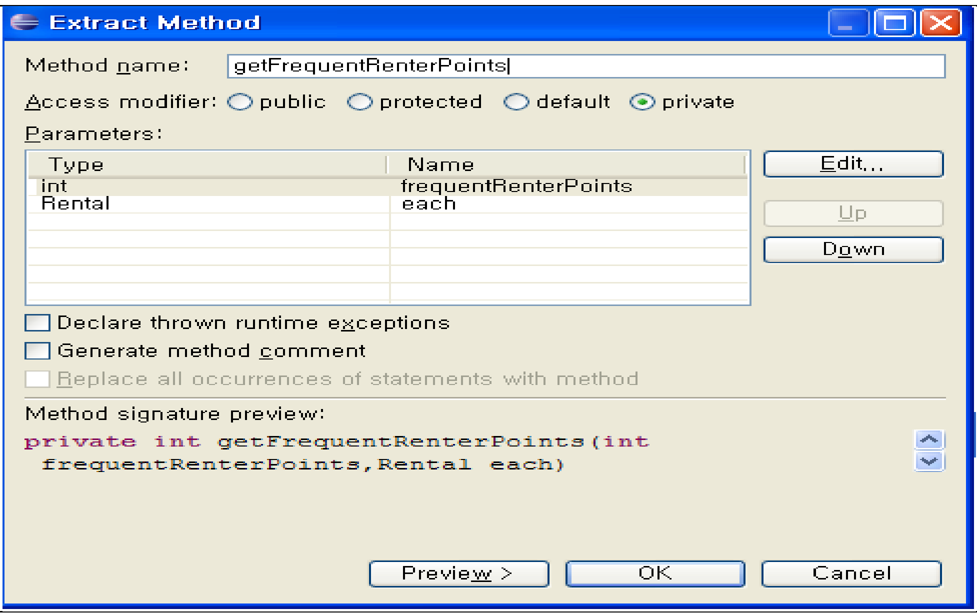
```
        public String statement() {
                double totalAmount = 0;
                int frequentRenterPoints = 0;
                String result = "Rental Record for " + getName() + "\n";
                Enumeration<Rental> rentalss = rentals.elements();
                while (rentalss.hasMoreElements()) {
                        Rental each = (Rental) rentalss.nextElement();
                        frequentRenterPoints ++;
                        if ( (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
                                each.getDayRented() > 1) frequentRenterPoints ++;

                        result += "\t" + each.getMovie().getTitle() + "\t" +
                                String.valueOf(each.getCharge()) + "\n";
```

```java
                    totalAmount += each.getCharge();

            }

            result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
            result += "You earned" + String.valueOf(frequentRenterPoints) +
                    " frequent renter points";
            return result;


    }
```

18. statement 메소드의 포인트 적립 부분을 별도의 메소드로 도출한다. 다음과 같이 포인트 적립 코드를 선택한다.

```java
    public String statement() {
            double totalAmount = 0;
            int frequentRenterPoints = 0;
            String result = "Rental Record for " + getName() + "\n";
            Enumeration<Rental> rentalss = rentals.elements();
            while (rentalss.hasMoreElements()) {
                    Rental each = (Rental) rentalss.nextElement();
                    frequentRenterPoints ++;
                    if ( (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
                                each.getDayRented() > 1) frequentRenterPoints ++;

                    result += "\t" + each.getMovie().getTitle() + "\t" +
                            String.valueOf(each.getCharge()) + "\n";
                    totalAmount += each.getCharge();

            }

            result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
            result += "You earned" + String.valueOf(frequentRenterPoints) +
                    " frequent renter points";
            return result;


    }
```

19. 오른쪽 마우스 버튼을 누르고 Refactor > Extract Method 를 선택한다. 메소드 이름에는 getFrequentRenterPoints를 넣는다.


Extract Method

Method name: getFrequentRenterPoints

Access modifier: ( ) public ( ) protected ( ) default [✓ private]

Parameters:

Type	Name
int	frequentRenterPoints
Rental	each

☐ Declare thrown runtime exceptions
☐ Generate method comment
☐ Replace all occurrences of statements with method

Method signature preview:

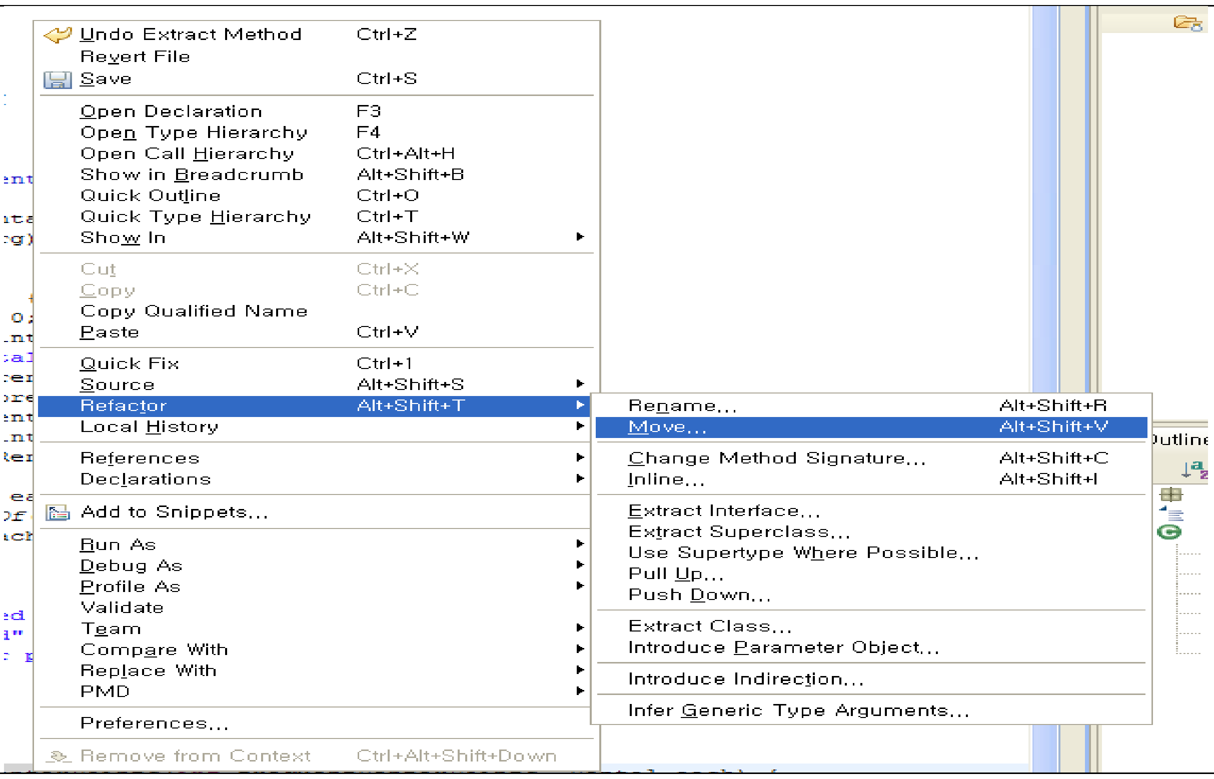
```
private int getFrequentRenterPoints(int
 frequentRenterPoints,Rental each)
```

20. OK 버튼을 누르면 Customer 클래스가 다음과 같이 변형된다.

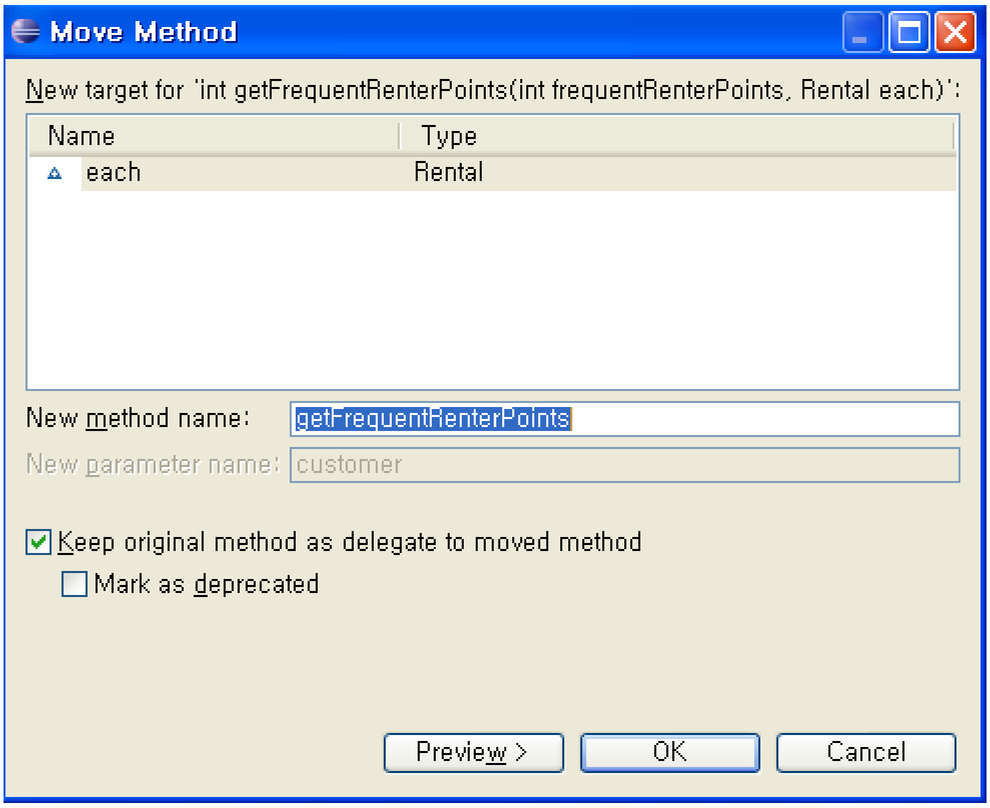
```
	public String statement() {
		double totalAmount = 0;
		int frequentRenterPoints = 0;
		String result = "Rental Record for " + getName() + "\n";
		Enumeration<Rental> rentalss = rentals.elements();
		while (rentalss.hasMoreElements()) {
			Rental each = (Rental) rentalss.nextElement();
			frequentRenterPoints = getFrequentRenterPoints(
					frequentRenterPoints, each);

			result += "\t" + each.getMovie().getTitle() + "\t" +
				String.valueOf(each.getCharge()) + "\n";
			totalAmount += each.getCharge();

		}

		result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
		result += "You earned" + String.valueOf(frequentRenterPoints) +
			" frequent renter points";
		return result;


	}

	private int getFrequentRenterPoints(int frequentRenterPoints, Rental each) {
		frequentRenterPoints ++;
		if ( (each.getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
				each.getDayRented() > 1) frequentRenterPoints ++;
		return frequentRenterPoints;
	}
```

21. getFrequentRenterPoints 메소드를 Rental 클래스로 이동한다.

getFrequentRenterPoints 메소드를 선택하고 오른 쪽 마우스 버튼을 누른다. Refactor > Move를 선택한다.



22. 다음과 같은 대화상자가 나타난다.


Move Method

New target for 'int getFrequentRenterPoints(int frequentRenterPoints, Rental each)':

Name	Type
each	Rental

New method name: getFrequentRenterPoints

New parameter name: customer

[✓] Keep original method as delegate to moved method

[ ] Mark as deprecated

23. OK를 누르면 Customer 클래스는 다음과 같이 변형된다.

```
public String statement() {
        double totalAmount = 0;
```

```
            int frequentRenterPoints = 0;
            String result = "Rental Record for " + getName() + "\n";
            Enumeration<Rental> rentalss = rentals.elements();
            while (rentalss.hasMoreElements()) {
                    Rental each = (Rental) rentalss.nextElement();
                    frequentRenterPoints = getFrequentRenterPoints(
                                    frequentRenterPoints, each);

                    result += "\t" + each.getMovie().getTitle() + "\t" +
                            String.valueOf(each.getCharge()) + "\n";
                    totalAmount += each.getCharge();

            }

            result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
            result += "You earned" + String.valueOf(frequentRenterPoints) +
                    " frequent renter points";
            return result;


    }

    private int getFrequentRenterPoints(int frequentRenterPoints, Rental each) {
            return each.getFrequentRenterPoints(frequentRenterPoints);
    }
```

24. Rental 클래스는 getFrequentRenterPoints 메소드가 추가된다.

```
    int getFrequentRenterPoints(int frequentRenterPoints) {
            frequentRenterPoints ++;
            if ( (getMovie().getPriceCode() == Movie.NEW_RELEASE) &&
                            getDayRented() > 1) frequentRenterPoints ++;
            return frequentRenterPoints;
    }
```

25. Customer 클래스에서 getFrequentRenterPoints 메소드를 제거하고 each.getFrequentRenterPoints를 직접 사용한다.

```
    public String statement() {
            double totalAmount = 0;
            int frequentRenterPoints = 0;
            String result = "Rental Record for " + getName() + "\n";
            Enumeration<Rental> rentalss = rentals.elements();
            while (rentalss.hasMoreElements()) {
                    Rental each = (Rental) rentalss.nextElement();
                    frequentRenterPoints =
each.getFrequentRenterPoints(frequentRenterPoints);

                    result += "\t" + each.getMovie().getTitle() + "\t" +
                            String.valueOf(each.getCharge()) + "\n";
                    totalAmount += each.getCharge();

            }

            result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
            result += "You earned" + String.valueOf(frequentRenterPoints) +
                    " frequent renter points";
            return result;


    }
```

26. getFrequentRenterPoints에 파라메터가 들어갈 필요가 없기 때문에 파라메터를 제거한다.

```
	public String statement() {
		double totalAmount = 0;
		int frequentRenterPoints = 0;
		String result = "Rental Record for " + getName() + "\n";
		Enumeration<Rental> rentalss = rentals.elements();
		while (rentalss.hasMoreElements()) {
			Rental each = (Rental) rentalss.nextElement();
			frequentRenterPoints += each.getFrequentRenterPoints();

			result += "\t" + each.getMovie().getTitle() + "\t" +
				String.valueOf(each.getCharge()) + "\n";
			totalAmount += each.getCharge();

		}

		result += "Amount owed is " + String.valueOf(totalAmount) + "\n";
		result += "You earned" + String.valueOf(frequentRenterPoints) +
			" frequent renter points";
		return result;


	}
```

27. Rental 클래스의 getFrequentRenterPoints 메소드는 NEW_RELEASE일 경우는 포인트가 2가 추가되고 아미년 1이 추가된다. 따라서 코드를 다음과 같이 단순하게 변형한다.

```
	int getFrequentRenterPoints() {

		if ( (getMovie().getPriceCode() == Movie.NEW_RELEASE)  &&
				getDayRented() > 1) return 2;
		return 1;
	}
```

28. RentalTest에 다음 메소드를 추가한다. NEW_RELEASE이기 때문에 포인트는 1이여야 한다.

```
	public void testGetFrequentRenterPoints(){
		Movie movie = new Movie(Movie.NEW_RELEASE,"스파이더맨");
		Rental rental;
		try {
			rental = new Rental(movie, 1);
			assertTrue(rental.getFrequentRenterPoints() == 1);
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
```

29. JUnit 테스트를 수행하여 결과를 확인한다.

30. statement 메소드에서 while 루프는 대여 결과를 출력하는 것 외에 총 대여금을 계산한다. 두가지 작업은 분리하는 것이 바람직하다. 따라서 총 대여금을 계산하는 메소드를 별도로 작성한다.

```
	public String statement() {
		double totalAmount = 0;
		int frequentRenterPoints = 0;
		String result = "Rental Record for " + getName() + "\n";
		Enumeration<Rental> rentalss = rentals.elements();
		while (rentalss.hasMoreElements()) {
			Rental each = (Rental) rentalss.nextElement();
			frequentRenterPoints += each.getFrequentRenterPoints();
```

```
                result += "\t" + each.getMovie().getTitle() + "\t" +
                        String.valueOf(each.getCharge()) + "\n";
        }

        result += "Amount owed is " + String.valueOf(getTotalCharge()) + "\n";
        result += "You earned" + String.valueOf(frequentRenterPoints) +
                " frequent renter points";
        return result;


}

private double getTotalCharge(){
        double result = 0;
        Enumeration<Rental> rentalss = rentals.elements();
        while (rentalss.hasMoreElements()) {
                Rental each = (Rental) rentalss.nextElement();
                result += each.getCharge();

        }
        return result;
}
```

32. 같은 방법으로 포인터의 총액을 구하는 메소드로 별도로 구현한다.

```
public String statement() {
        double totalAmount = 0;
        int frequentRenterPoints = 0;
        String result = "Rental Record for " + getName() + "\n";
        Enumeration<Rental> rentalss = rentals.elements();
        while (rentalss.hasMoreElements()) {
                Rental each = (Rental) rentalss.nextElement();

                result += "\t" + each.getMovie().getTitle() + "\t" +
                        String.valueOf(each.getCharge()) + "\n";
        }

        result += "Amount owed is " + String.valueOf(getTotalCharge()) + "\n";
        result += "You earned" + String.valueOf(getTotalFrequentRenterPoints()) +
                " frequent renter points";
        return result;


}

private double getTotalCharge(){
        double result = 0;
        Enumeration<Rental> rentalss = rentals.elements();
        while (rentalss.hasMoreElements()) {
                Rental each = (Rental) rentalss.nextElement();
                result += each.getCharge();

        }
        return result;
}

private int getTotalFrequentRenterPoints(){
        int result = 0;
        Enumeration<Rental> rentalss = rentals.elements();
```

```
                while (rentalss.hasMoreElements()) {
                        Rental each = (Rental) rentalss.nextElement();
                        result += each.getFrequentRenterPoints();

                }
                return result;
        }
```

조건문을 폴리포피즘을 사용하여 변형함

1. Rental 클래스 getCharge를 Movie 클래스로 옮긴다. (Refactor > Move 메뉴를 사용함)

2. Movie 클래스의 getCharge는 다음과 같다.

```
        double getCharge(Rental rental) {
                double result = 0;
                switch (rental.getMovie().getPriceCode()) {
                        case Movie.REGULAR:
                                result += 2;
                                if (rental.getDayRented() > 2)
                                        result += (rental.getDayRented() - 2) * 1.5;
                                break;
                        case Movie.NEW_RELEASE:
                                result += rental.getDayRented() * 3;
                                break;
                        case Movie.CHILDRENS:
                                result += 1.5;
                                if (rental.getDayRented() > 3)
                                        result += (rental.getDayRented() - 3) * 1.5;
                }
                return result;
        }
```

3. 파라메터를 rental를 받는 것보다는 int daysRented를 받는 것이 바람직하기 때문에 다음과 같이 변형한다.

```
        double getCharge(int daysRented) {
                double result = 0;
                switch (getPriceCode()) {
                        case Movie.REGULAR:
                                result += 2;
                                if (daysRented > 2)
                                        result += (daysRented - 2) * 1.5;
                                break;
                        case Movie.NEW_RELEASE:
                                result += daysRented * 3;
                                break;
                        case Movie.CHILDRENS:
                                result += 1.5;
                                if (daysRented > 3)
                                        result += (daysRented - 3) * 1.5;
                }
                return result;
        }
```

4. Rental 클래스의 getCharge는 다음과 같이 변형한다.

```
        double getCharge() {
                return movie.getCharge(dayRented);
        }
```

5. 다음과 같은 PriceCode에 관련된 클래스를 새로운 파일로 만들어서 추가한다.

```
package com.videorent;
```

```java
abstract class Price {
	abstract int getPriceCode();
}

class ChildrenPrice extends Price{
	int getPriceCode(){
		return Movie.CHILDRENS;
	}
}

class NewReleasePrice extends Price{
	int getPriceCode(){
		return Movie.NEW_RELEASE;
	}
}


class RegularPrice extends Price{
	int getPriceCode(){
		return Movie.REGULAR;
	}
}
```

6. Movie 클래스에서 지역변수 priceCode를 제거하고 Price 클래스의 인스턴스로 대처한다.

```java
public class Movie {

	public static final int CHILDRENS = 2;
	public static final int NEW_RELEASE = 1;
	public static final int REGULAR = 0;
	private Price price;

	private String title;

	public Movie(int priceCode, String title) {
		this.setPriceCode(priceCode);
		this.title = title;
	}

	public int getPriceCode() {
		return price.getPriceCode();
	}

	public void setPriceCode(int arg){
		switch (arg) {
		case REGULAR:
			price = new RegularPrice();
			break;
		case CHILDRENS:
			price = new ChildrenPrice();
			break;
		case NEW_RELEASE:
			price = new NewReleasePrice();
			break;
		default:
			throw new IllegalArgumentException("Incorrent Price Code");
		}
	}
```

7. getCharge 메소드 내부의 로직을 Price 클래스로 이동한다.(Extract Method와 Move 사용)

Move class

```
	double getCharge(int daysRented) {
		return price.getCharge(daysRented);
	}
```

Price Class

```
abstract class Price {
	abstract int getPriceCode();

	double getCharge(int daysRented) {
		double result = 0;
		switch (getPriceCode()) {
			case Movie.REGULAR:
				result += 2;
				if (daysRented > 2)
					result += (daysRented - 2) * 1.5;
				break;
			case Movie.NEW_RELEASE:
				result += daysRented * 3;
				break;
			case Movie.CHILDRENS:
				result += 1.5;
				if (daysRented > 3)
					result += (daysRented - 3) * 1.5;
		}
		return result;
	}
}
```

8. Price Class의 getCharge를 abstract method로 만들고 하위 클래스에 getCharge 메소드를 만든다.

```
package com.videorent;

abstract class Price {
	abstract int getPriceCode();

	abstract double getCharge(int daysRented);
}

class ChildrenPrice extends Price{
	int getPriceCode(){
		return Movie.CHILDRENS;
	}

	double getCharge(int daysRented) {
		double result = 0;
		result += 1.5;
		if (daysRented > 3)
			result += (daysRented - 3) * 1.5;
		return result;
	}
}

class NewReleasePrice extends Price{
	int getPriceCode(){
		return Movie.NEW_RELEASE;
	}

	double getCharge(int daysRented) {
		double result = 0;
		result += daysRented * 3;
```

```
            return result;
        }
}


class RegularPrice extends Price{
        int getPriceCode(){
                return Movie.REGULAR;
        }

        double getCharge(int daysRented) {
                double result = 0;
                result += 2;
                if (daysRented > 2)
                        result += (daysRented - 2) * 1.5;
                return result;
        }
}
```

9. CustomerTest 클래스를 만들고 다음 메소드를 추가하여 프로그램이 제대로 동작하는지 살펴본다.

```
package com.videorent;

import junit.framework.TestCase;

public class CustmerTest extends TestCase {

        public void testStatement() {
                Customer customer = new Customer();
                Rental rental1, rental2;
                try {
                        Movie movie1 = new Movie(Movie.NEW_RELEASE,"스파이더맨");

                        Movie movie2 = new Movie(Movie.CHILDRENS,"짱구");
                        rental1 = new Rental(movie1, 5);
                        rental2 = new Rental(movie2, 7);
                        customer.addRental(rental1);
                        customer.addRental(rental2);
                        assertTrue(customer.getTotalCharge() == 22.5);
                        System.out.println(customer.statement());
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

}
```