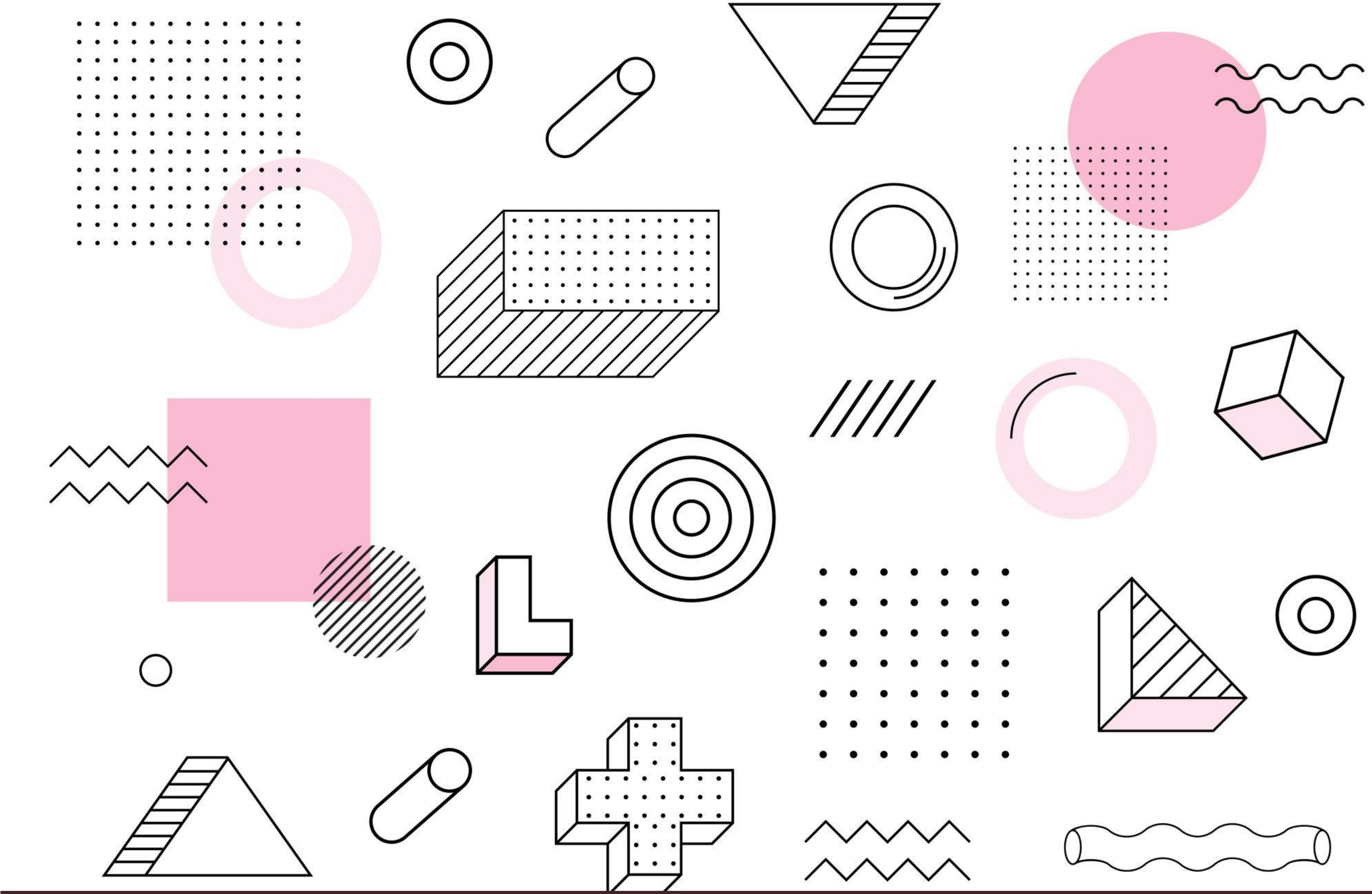


1학기 고급자바 실습

week 3-2

김민진(18)

김지희(18)

문의 메일 : genie02166@duksung.ac.kr

Part1. 제네릭

1) 제네릭 타입

2) 멀티 타입 파라미터

3) 제네릭 메소드

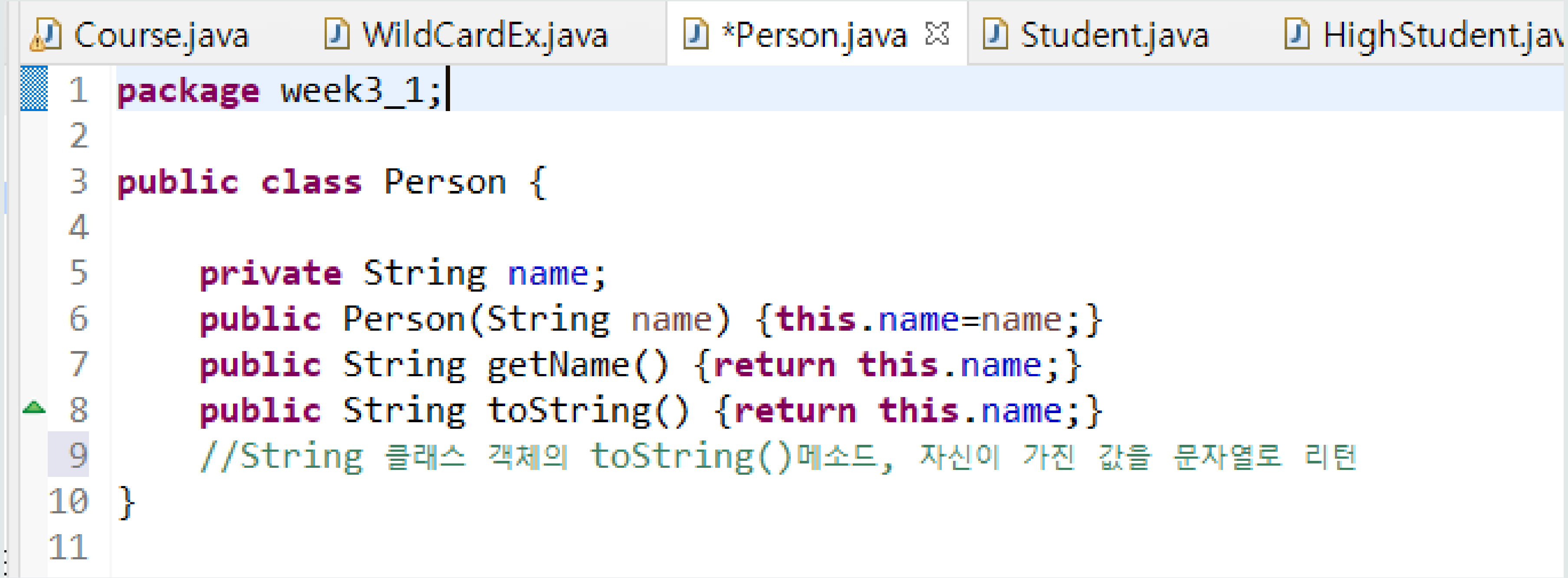
- 예제 1
- 예제 2

4) 제한된 타입 파라미터

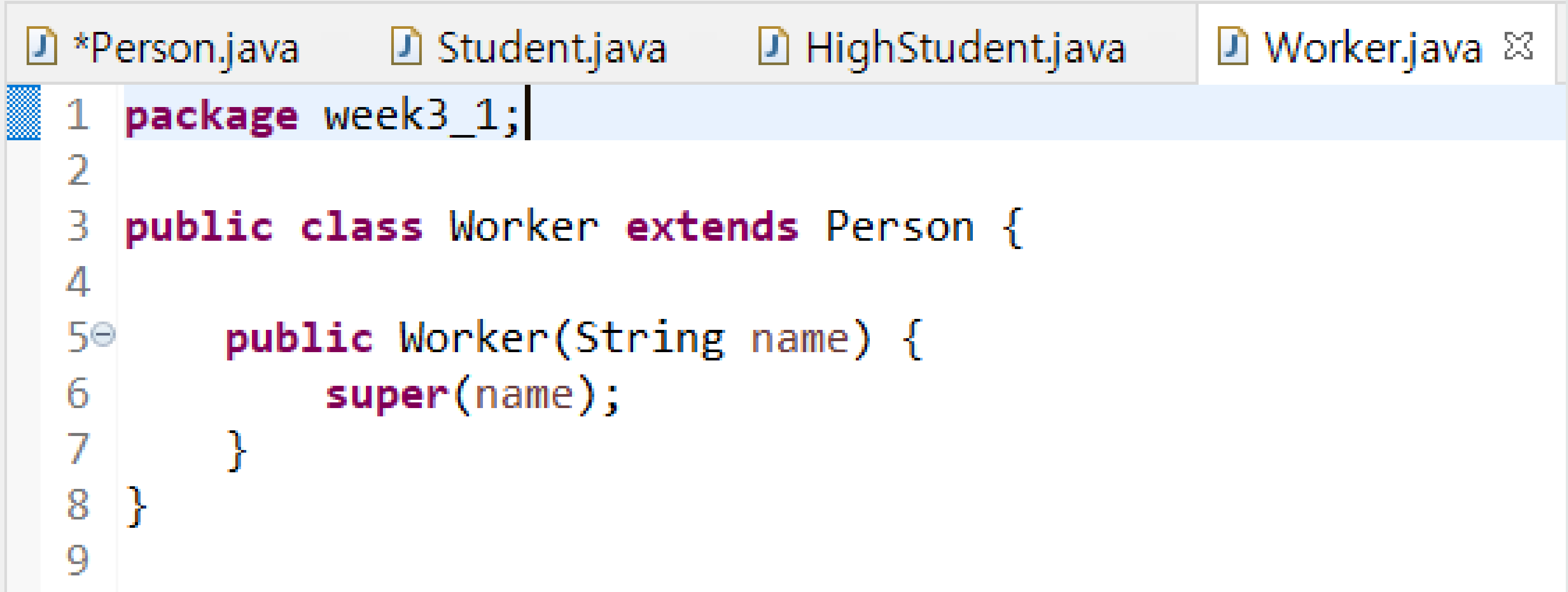
5) 와일드카드 타입

6) 제네릭 타입의 상속과 구현

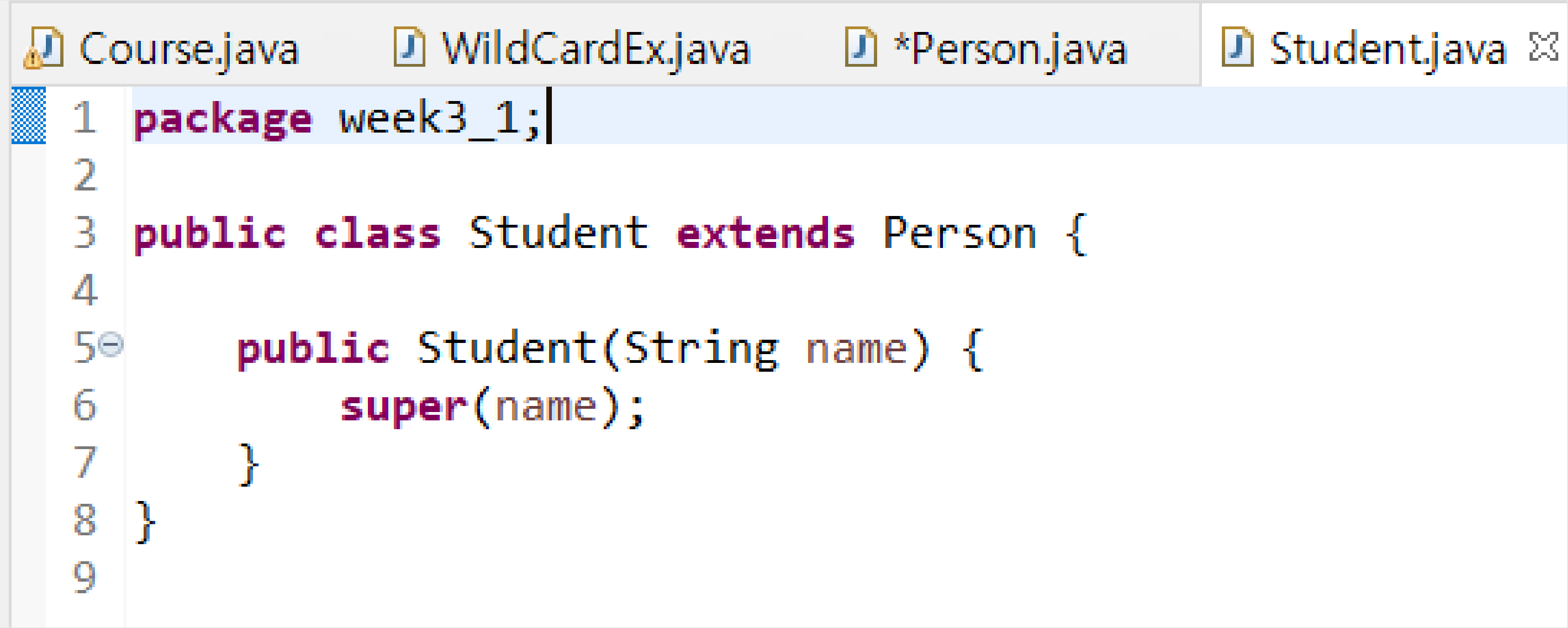
5) 와일드카드 타입



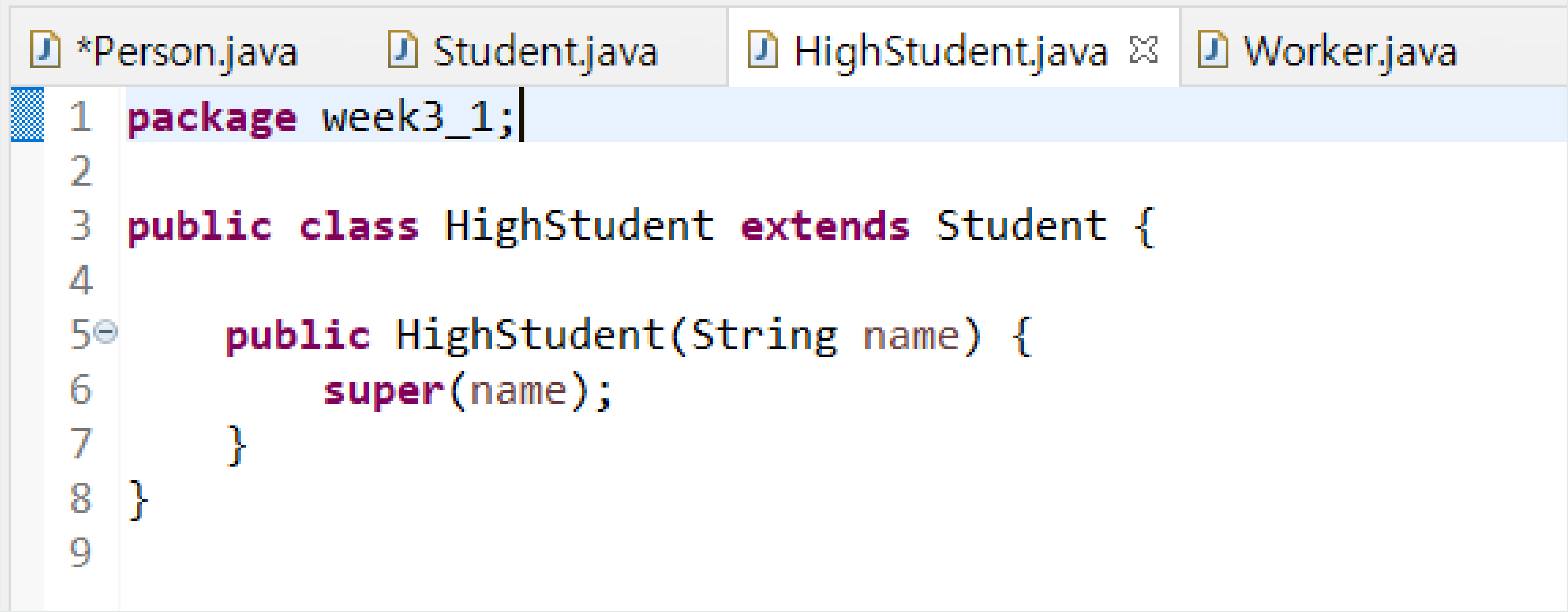
```java
package week3_1;

public class Person {

    private String name;
    public Person(String name) {this.name=name;}
    public String getName() {return this.name;}
    public String toString() {return this.name;}
    //String 클래스 객체의 toString()메소드, 자신이 가진 값을 문자열로 리턴
}

```



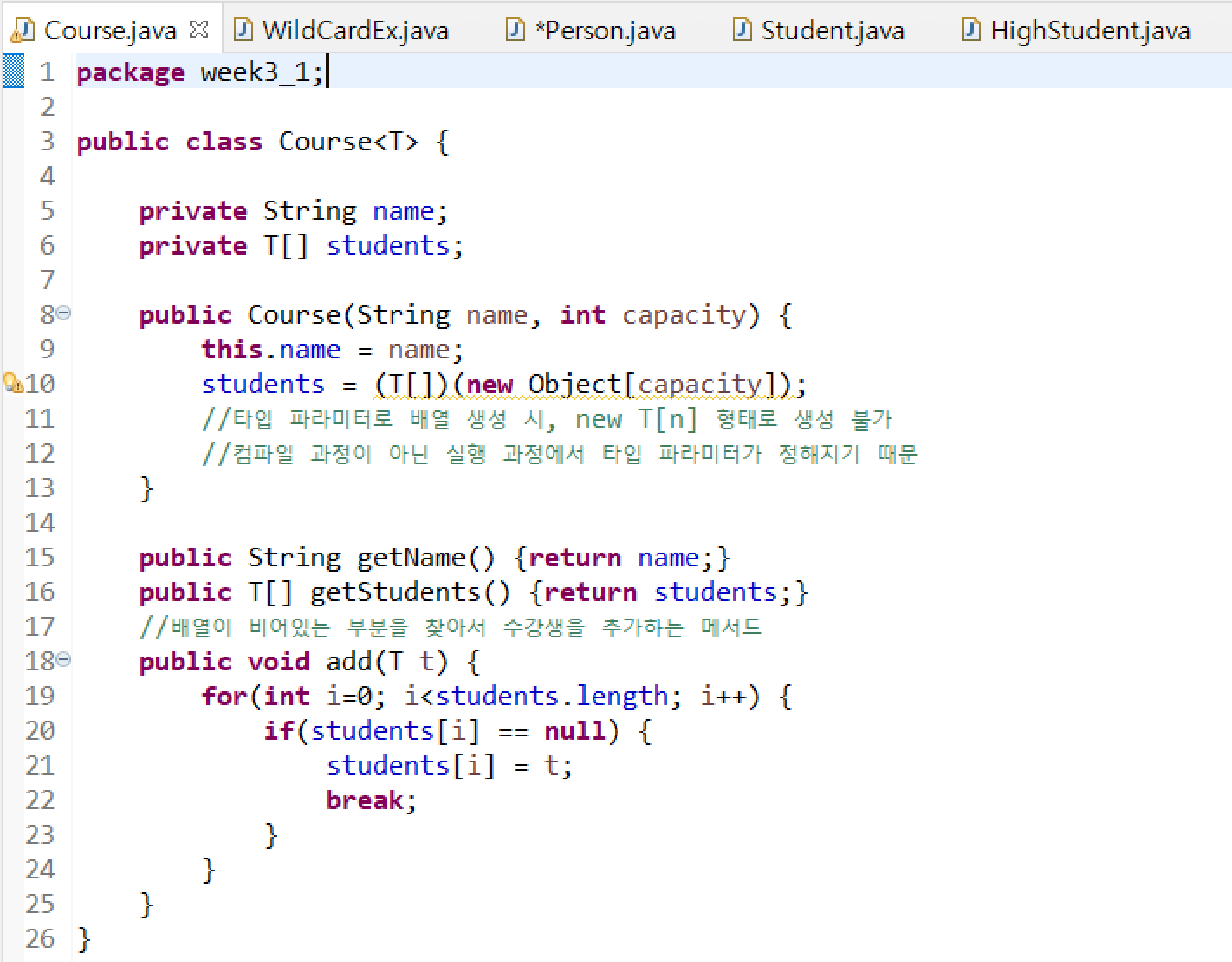
```java
package week3_1;

public class Worker extends Person {

    public Worker(String name) {
        super(name);
    }
}

```



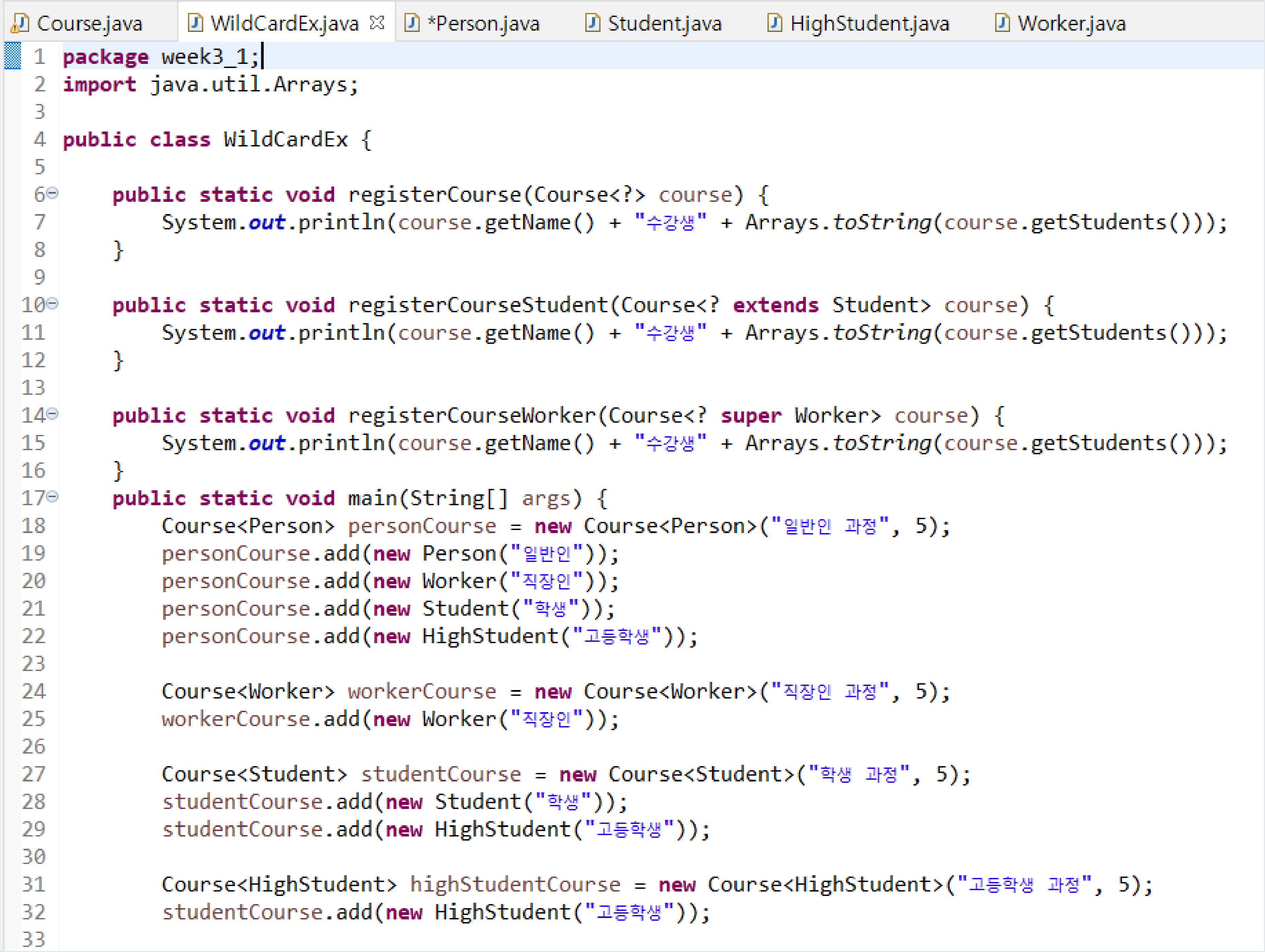
```java
package week3_1;

public class Student extends Person {

    public Student(String name) {
        super(name);
    }
}

```



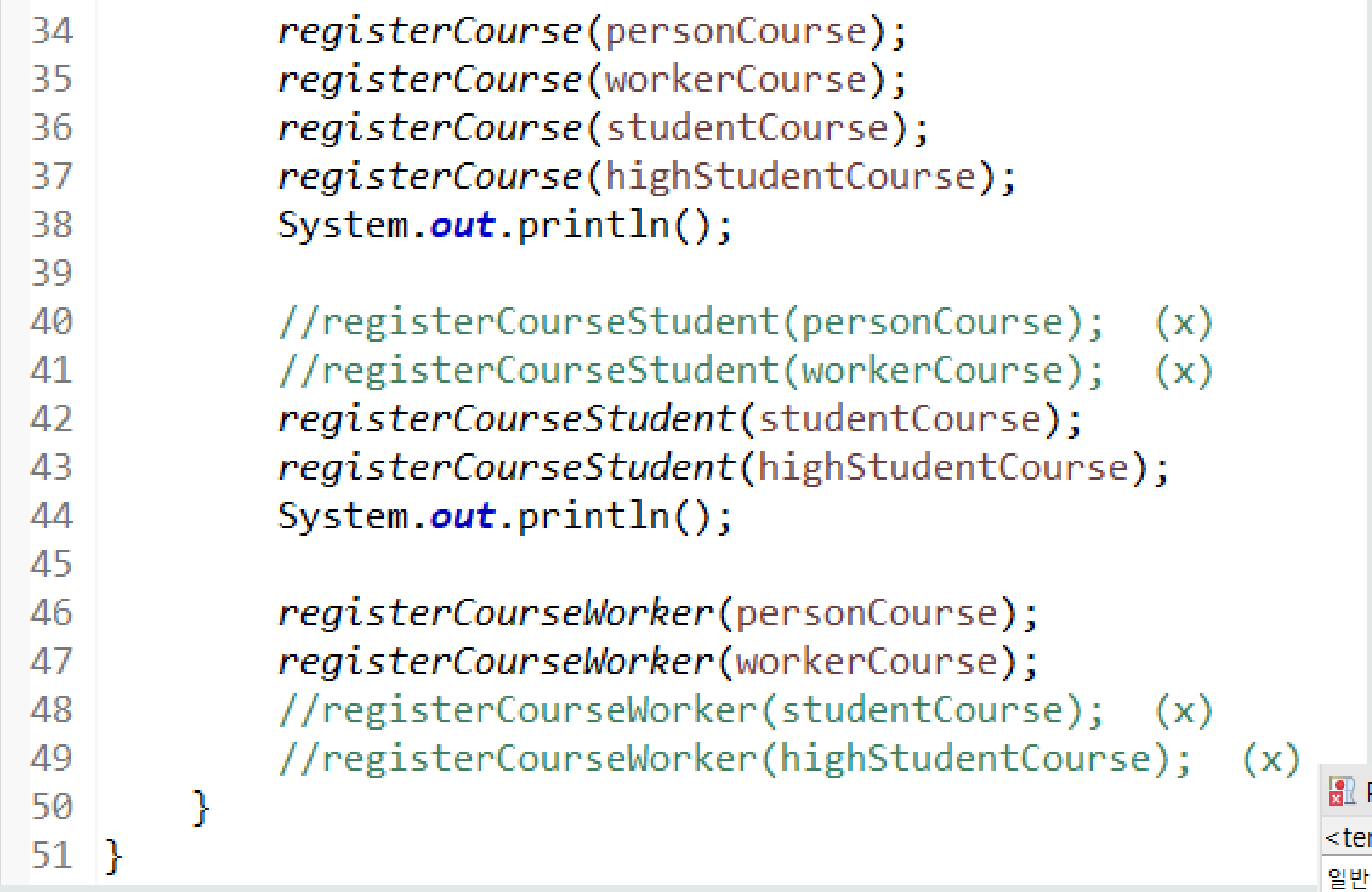
```java
package week3_1;

public class HighStudent extends Student {

    public HighStudent(String name) {
        super(name);
    }
}

```



```java
package week3_1;

public class Course<T> {

    private String name;
    private T[] students;

    public Course(String name, int capacity) {
        this.name = name;
        students = (T[])(new Object[capacity]);
        //타입 파라미터로 배열 생성 시, new T[n] 형태로 생성 불가
        //컴파일 과정이 아닌 실행 과정에서 타입 파라미터가 정해지기 때문
    }

    public String getName() {return name;}
    public T[] getStudents() {return students;}
    //배열이 비어있는 부분을 찾아서 수강생을 추가하는 메서드
    public void add(T t) {
        for(int i=0; i<students.length; i++) {
            if(students[i] == null) {
                students[i] = t;
                break;
            }
        }
    }
}
```

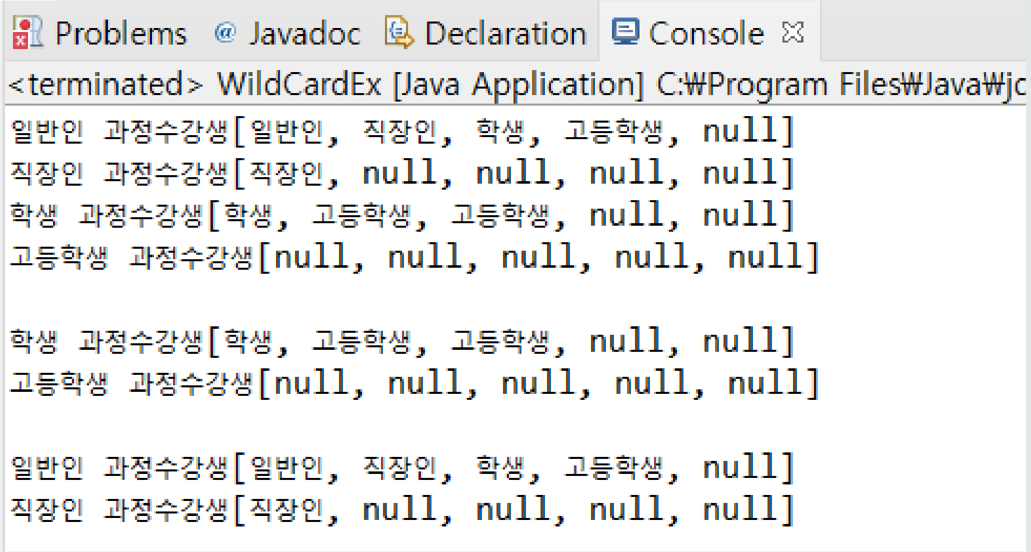


```
package week3_1;
import java.util.Arrays;

public class WildCardEx {

    public static void registerCourse(Course<?> course) {
        System.out.println(course.getName() + "수강생" + Arrays.toString(course.getStudents()));
    }

    public static void registerCourseStudent(Course<? extends Student> course) {
        System.out.println(course.getName() + "수강생" + Arrays.toString(course.getStudents()));
    }

    public static void registerCourseWorker(Course<? super Worker> course) {
        System.out.println(course.getName() + "수강생" + Arrays.toString(course.getStudents()));
    }
    public static void main(String[] args) {
        Course<Person> personCourse = new Course<Person>("일반인 과정", 5);
        personCourse.add(new Person("일반인"));
        personCourse.add(new Worker("직장인"));
        personCourse.add(new Student("학생"));
        personCourse.add(new HighStudent("고등학생"));

        Course<Worker> workerCourse = new Course<Worker>("직장인 과정", 5);
        workerCourse.add(new Worker("직장인"));

        Course<Student> studentCourse = new Course<Student>("학생 과정", 5);
        studentCourse.add(new Student("학생"));
        studentCourse.add(new HighStudent("고등학생"));

        Course<HighStudent> highStudentCourse = new Course<HighStudent>("고등학생 과정", 5);
        studentCourse.add(new HighStudent("고등학생"));

```



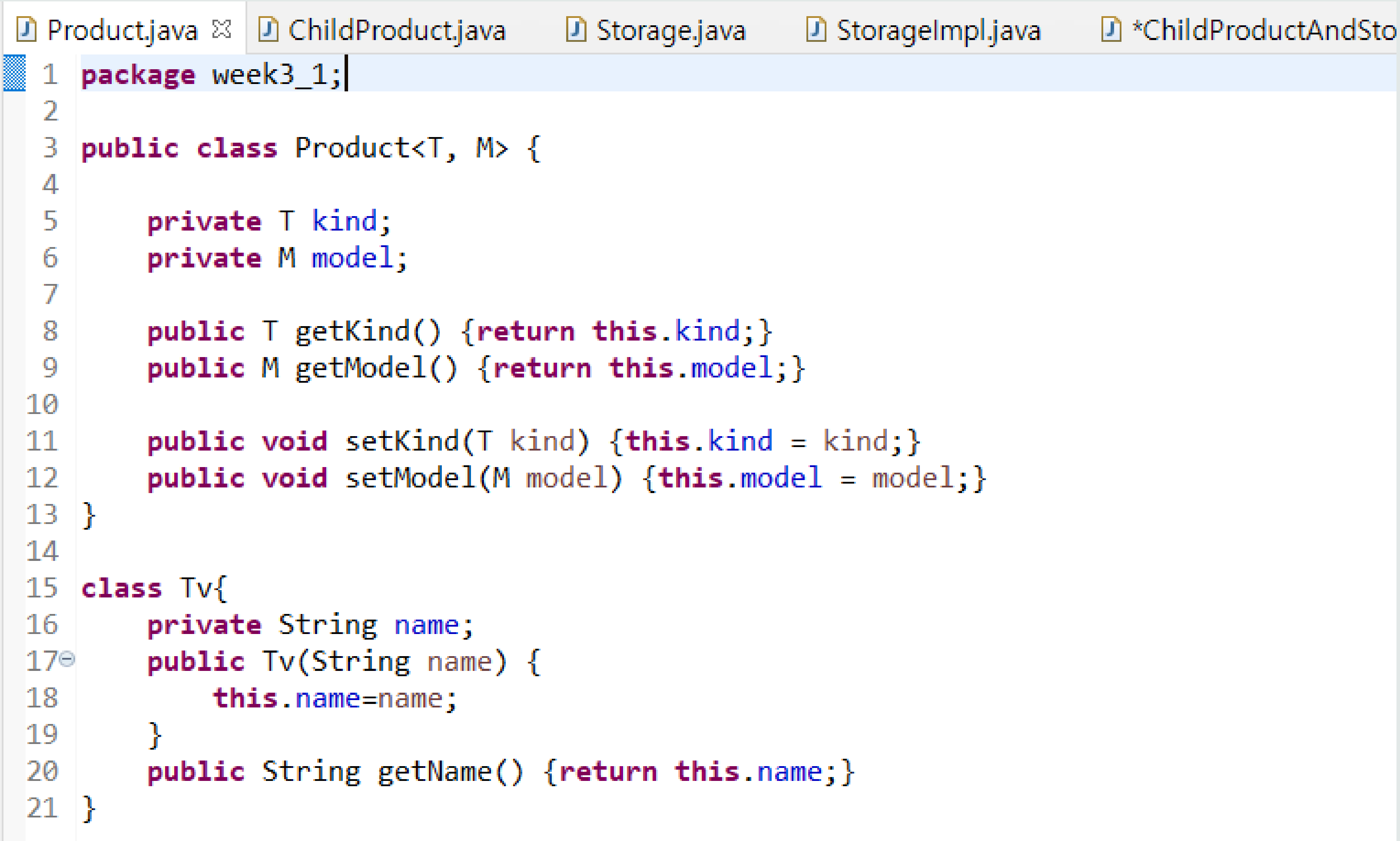
```java
        registerCourse(personCourse);
        registerCourse(workerCourse);
        registerCourse(studentCourse);
        registerCourse(highStudentCourse);
        System.out.println();

        //registerCourseStudent(personCourse);  (x)
        //registerCourseStudent(workerCourse);  (x)
        registerCourseStudent(studentCourse);
        registerCourseStudent(highStudentCourse);
        System.out.println();

        registerCourseWorker(personCourse);
        registerCourseWorker(workerCourse);
        //registerCourseWorker(studentCourse);  (x)
        //registerCourseWorker(highStudentCourse);  (x)
    }
}
```

실행 결과

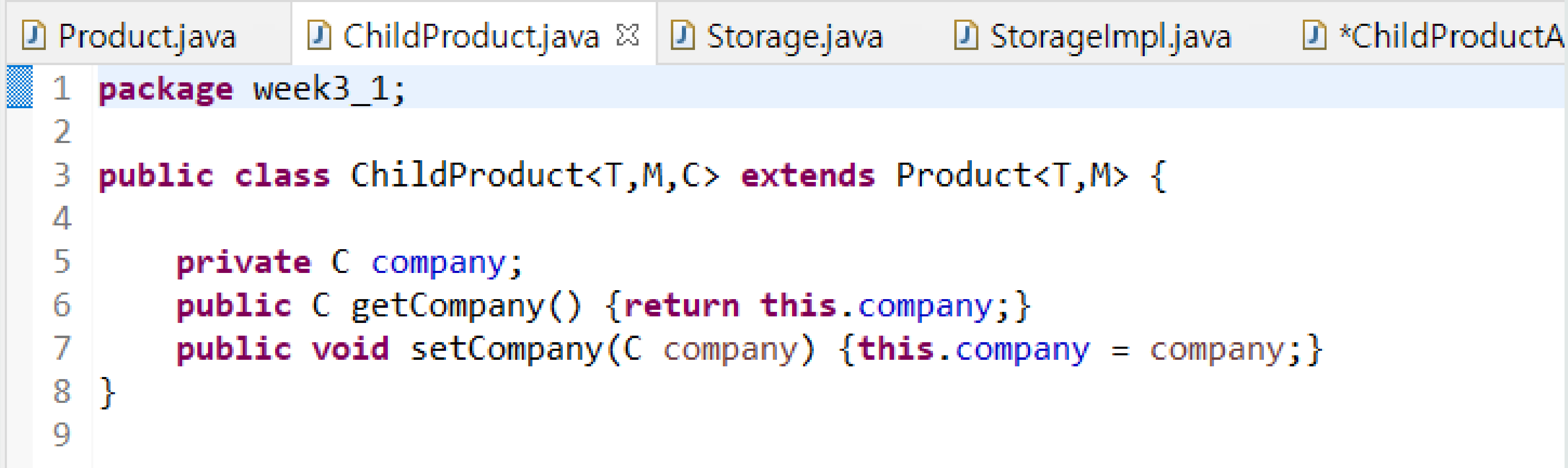


```
<terminated> WildCardEx [Java Application] C:₩Program Files₩Java₩jc
일반인 과정수강생[일반인, 직장인, 학생, 고등학생, null]
직장인 과정수강생[직장인, null, null, null, null]
학생 과정수강생[학생, 고등학생, 고등학생, null, null]
고등학생 과정수강생[null, null, null, null, null]

학생 과정수강생[학생, 고등학생, 고등학생, null, null]
고등학생 과정수강생[null, null, null, null, null]

일반인 과정수강생[일반인, 직장인, 학생, 고등학생, null]
직장인 과정수강생[직장인, null, null, null, null]
```

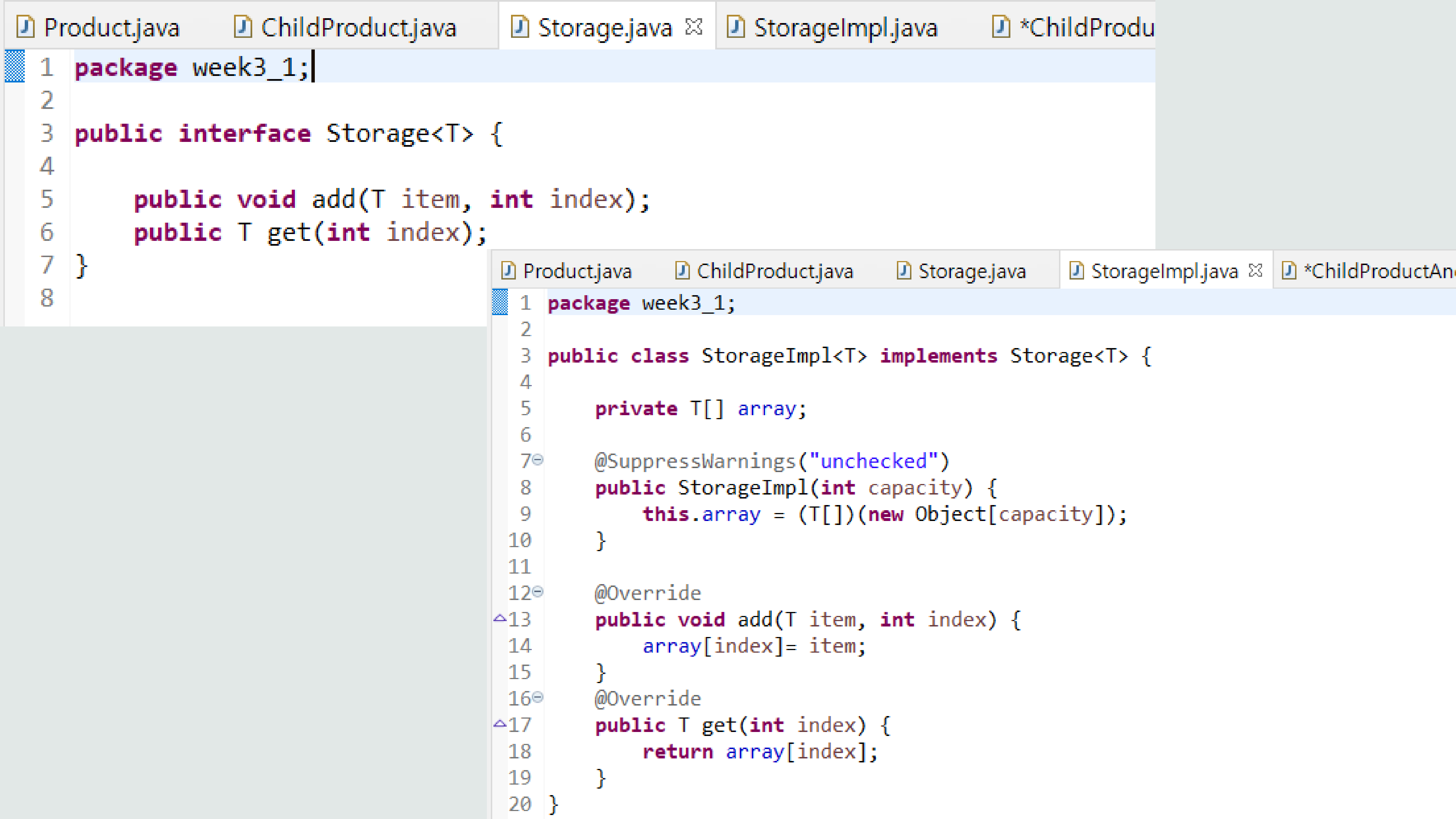
6) 제네릭 타입의 상속과 구현



```java
package week3_1;

public class Product<T, M> {

    private T kind;
    private M model;

    public T getKind() {return this.kind;}
    public M getModel() {return this.model;}

    public void setKind(T kind) {this.kind = kind;}
    public void setModel(M model) {this.model = model;}
}

class Tv{
    private String name;
    public Tv(String name) {
        this.name=name;
    }
    public String getName() {return this.name;}
}
```

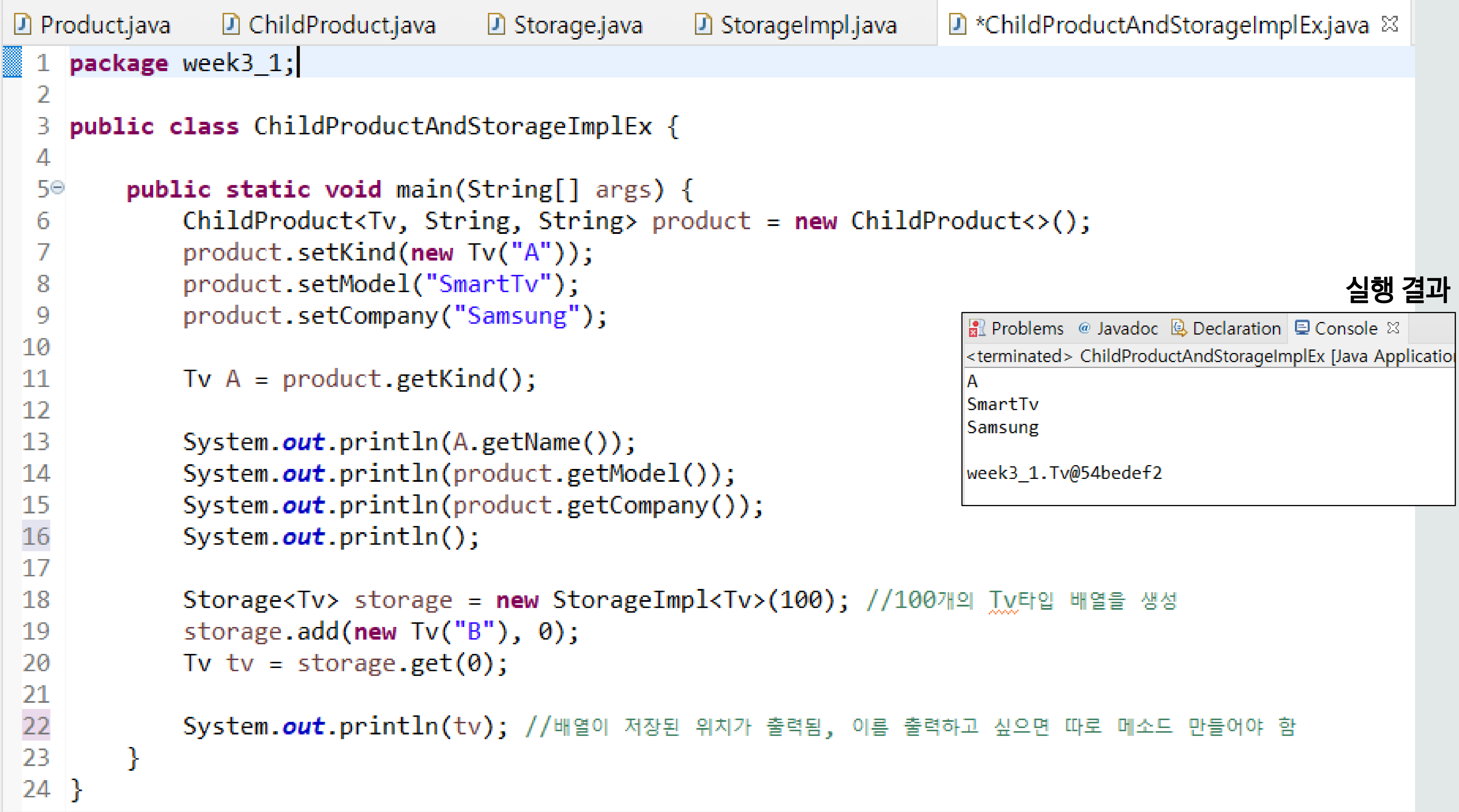


```java
package week3_1;

public class ChildProduct<T,M,C> extends Product<T,M> {

    private C company;
    public C getCompany() {return this.company;}
    public void setCompany(C company) {this.company = company;}
}

```



```java
package week3_1;

public interface Storage<T> {

    public void add(T item, int index);
    public T get(int index);
}

```

```java
package week3_1;

public class StorageImpl<T> implements Storage<T> {

    private T[] array;

    @SuppressWarnings("unchecked")
    public StorageImpl(int capacity) {
        this.array = (T[])(new Object[capacity]);
    }

    @Override
    public void add(T item, int index) {
        array[index]= item;
    }
    @Override
    public T get(int index) {
        return array[index];
    }
}
```



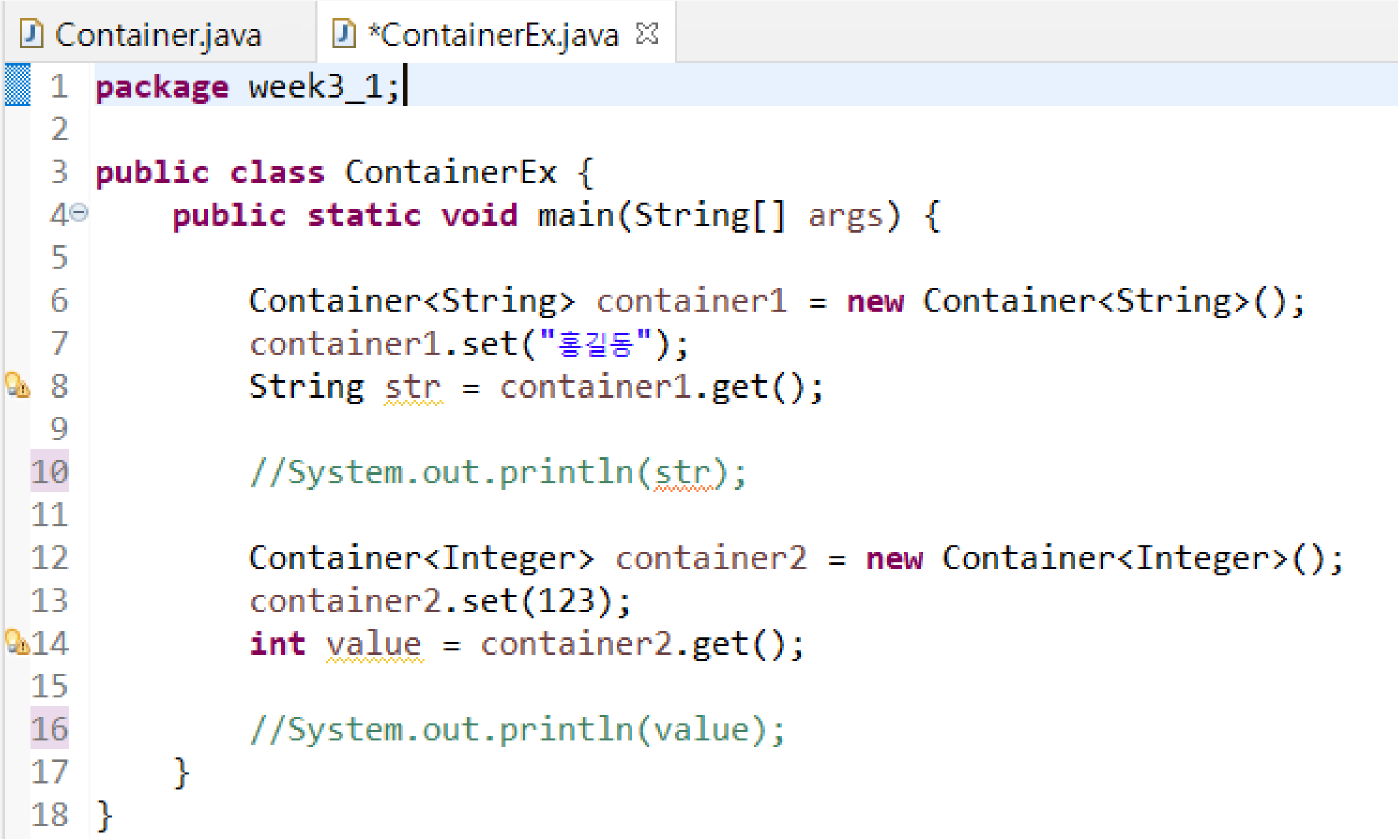
```java
package week3_1;

public class ChildProductAndStorageImplEx {

    public static void main(String[] args) {
        ChildProduct<Tv, String, String> product = new ChildProduct<>();
        product.setKind(new Tv("A"));
        product.setModel("SmartTv");
        product.setCompany("Samsung");

        Tv A = product.getKind();

        System.out.println(A.getName());
        System.out.println(product.getModel());
        System.out.println(product.getCompany());
        System.out.println();

        Storage<Tv> storage = new StorageImpl<Tv>(100); //100개의 Tv타입 배열을 생성
        storage.add(new Tv("B"), 0);
        Tv tv = storage.get(0);

        System.out.println(tv); //배열이 저장된 위치가 출력됨, 이름 출력하고 싶으면 따로 메소드 만들어야 함
    }
}
```

실행 결과

```
<terminated> ChildProductAndStorageImplEx [Java Applicatio
A
SmartTv
Samsung

week3_1.Tv@54bedef2
```

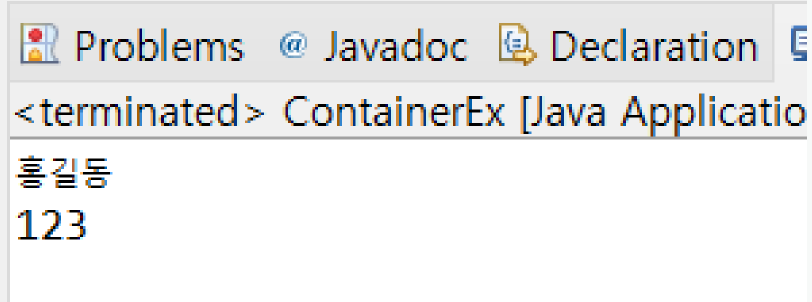
출석 과제 (4/2 금 오후 11:55 마감)

Q. ContainerEx의 실행을 가능하게 하는 Container 클래스를 작성해주세요. 실행 결과 창은 출력 메소드를 사용했을 경우의 결과입니다.



```java
package week3_1;

public class ContainerEx {
    public static void main(String[] args) {

        Container<String> container1 = new Container<String>();
        container1.set("홍길동");
        String str = container1.get();

        //System.out.println(str);

        Container<Integer> container2 = new Container<Integer>();
        container2.set(123);
        int value = container2.get();

        //System.out.println(value);
    }
}
```

실행 결과



```
<terminated> ContainerEx [Java Applicatio
홍길동
123
```