2019 소프트웨어프로젝트

Project 2

Java 및 *C* 프로그램 스타일 비교

중앙대학교

창의ICT공과대학 소프트웨어학부

20185659 김혜성

목차

Table of contents

1. Abstract

이 프로젝트에서는 C프로그램 스타일과 Java 프로그램 스타일의 차이를 체득하고, 실전적 경험을 통해 Java 프로그램의 장단점을 예측하며, Java의 장점을 살리는 프로그래밍 방식에 대한 목표 의식을 설정하는데 목적을 두어 'Java 및 C 프로그램 스타일 비교'를 진행하였다.

프로젝트 수행 결과는 다음과 같다.

1. **C**는 매개변수의 참조 방식이 **call by value**와 **call by reference**로 두가지 방식이 있지만**, Java**는 **call by value**만 존재한다**.**
2. **C**에서는 **structure**를 사용하지만 **Java**에서는 **class**를 사용한다**.**
3. **C**의 디자인은 절차적이지만 **Java**는 객체와 그 기능들이 중점이 된다**.**

1. **Java**는 가독성과 네이밍의 측면에서 **clean code**를 구현하기 쉽다**.**
2. **Java**는 코드의 재사용성이 강하다**.**
3. **Java**는 유지보수가 쉽다**.**
4. **Java**는 대형 프로젝트에 유리하다**.**

이 paper에서 프로젝트에 대한 고찰은 다음과 같다.

Java의 장점을 살리는 프로그래밍 방식은 Object-Oriented Programming이다. 즉, 객체지향적 디자인 설계가 **Java**의 장점을 극대화하는 방안이다**. Java** 프로그래밍을 하기 전에 객체 디자인 설계를 통해 기능 중심의 구현을 해야 한다**.**
이때 모든 언어의 프로그래밍에서 중요한 디자인은 **Top-down design**이다**. Top-down design**을 함으로써 **clean-code**의구현을 극대화하여 프로젝트에 이점을 줄 것이다**.**

2. Introduction

이 프로젝트의 목적은 다음과 같다.

1. **C** 프로그램 스타일과 **Java** 프로그램 스타일의 차이를 체득한다**.**
2. 실전적 경험을 통해 **Java** 프로그램의 장단점을 예측한다**.**
3. **Java**의 장점 살리는 프로그래밍 방식에 대한 목표 의식을 설정한다**.**

프로젝트의 내용은 다음과 같다.

1. 주어진 **C** 프로그램(**rectangle.c**)과 동일한 직사각형 관련 프로그램을 **Java** 스타일에 맞는 **Java** 프로그램으로 작성한다**.**
2. 두 프로그램을 세부비교 항목을 자체적으로 설정하여 두 프로그램을 비교한다**.**

3. Proposed Program

3.1. Analyzation of the program written in C

```c
#include <stdio.h>

struct rectangle {
   float width;
   float height;
} ;

void print(struct rectangle r) {
   printf("width = %f , height = %f \n", r.width, r.height);
}

bool equal(struct rectangle r1, struct rectangle r2) {
   if ( (r1.width == r2.width) && (r1.height == r2.height) )
      return true;
   else
      return false;
}

float compare(struct rectangle r1, struct rectangle r2) {
   return r1.width*r1.height - r2.width*r2.height;
}


void resize(struct rectangle *r, float ratio) {
   r->width *= ratio;
   r->height *= ratio;
}


main() {

   struct rectangle r1 = {1.0, 1.0};
   struct rectangle r2 = {1.0, 2.0};
   struct rectangle r3 = {2.0, 1.0};

   if ( equal(r2, r3) )
      printf("two rectangles are equal. \n");
   else
      printf("two rectangles are different. \n");

   if ( compare(r2, r3) > 0 )
      printf("the first is larger than the second. \n");
   else if ( compare(r2, r3) < 0 )
      printf("the first is smaller than the second. \n");
   else
      printf("two are the same in area. \n");

   print(r1);
   resize (&r1, 2.0);
   print(r1);

}
```

▲ **Fig. 1.** Rectangle Program written in C

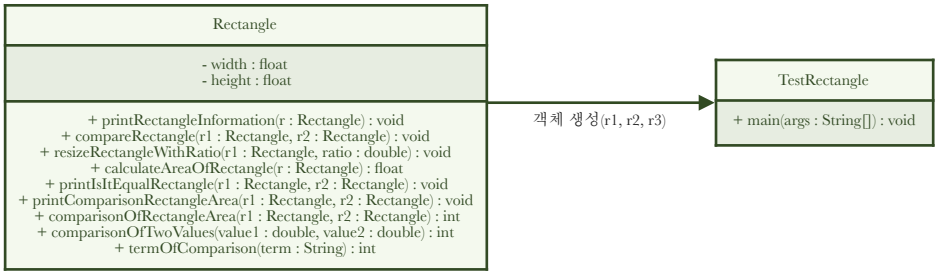
3-6 : rectangle 구조체 정의
8-10 : 매개변수로 전달된 r의 정보를 출력하는 print 함수 정의
12-17 : 매개변수로 전달된 r1과 r2가 같은지 비교하는 equal 함수 정의
19-21 : 매개변수로 전달된 r1과 r2의 넓이를 비교하는 compare 함수 정의
24-27 : 매개변수로 전달된 r의 주솟값에 있는 값을 변경하는 resize 함수 정의(call by reference)

30-52 : main 함수에서의 동작

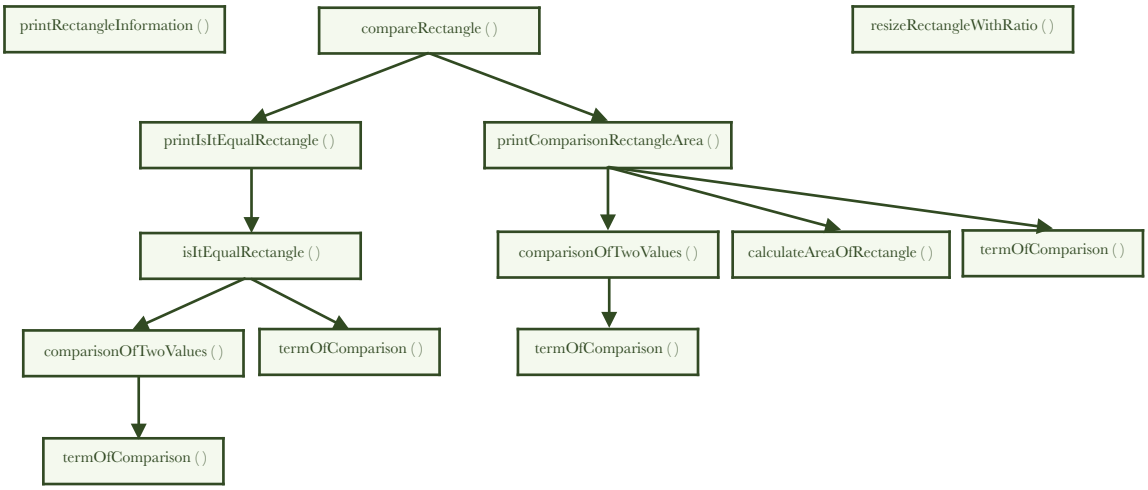
- 32-34 : rectangle 구조체 변수 선언 : r1, r2, r3
- 36-39 : r2와 r3이 같은지 equal 함수 호출로 비교
- 41-46 : r2와 r3의 넓이가 같은지 compare 함수 호출로 비교
- 48 : 수정 전 r1의 정보를 print 함수 호출로 출력
- 49 : r1의 정보를 resize 함수 호출로 수정(call by reference)
- 50 : 수정 후 r1의 정보를 print 함수 호출로 출력

3.2. Analyzation of the program written in Java

3.2.1. Design of the program



▲ **Fig. 2.** Design of the program written in Java



▲ **Fig. 3.** Design of the program written in Java

3.2.2. Source Code of the program

```java
package project2;

class Rectangle {
	private float width;
	private float height;

	Rectangle(double width, double height) {
		this.width = (float)width;
		this.height = (float)height;
	}

	void printRectangleInformation(Rectangle r) {
		System.out.println("Location of the Rectangle : " + r);
		System.out.println("Width of the Rectangle : " + r.width);
		System.out.println("Height of the Rectangle : " + r.height);
	}

	void compareRectangle(Rectangle r1, Rectangle r2) {
		printIsItEqualRectangle(r1, r2);
		printComparisonRectangleArea(r1, r2);
	}

	void resizeRectangleWithRatio(Rectangle r1, double ratio) {
		r1.width *= ratio;
		r1.height *= ratio;
	}

	float calculateAreaOfRectangle(Rectangle r) {
		return r.width * r.height;
	}

	void printIsItEqualRectangle(Rectangle r1, Rectangle r2) {
		if (isItEqualRectangle(r1, r2)) {
			System.out.println("Two rectangles are equal.");
		}
		else {
			System.out.println("Two rectangles are different.");
		}
	}

	void printComparisonRectangleArea(Rectangle r1, Rectangle r2) {

		if (comparisonOfRectangleArea(r1, r2) == termOfComparison("value1IsBigger")) {
			System.out.println("The Area of the first rectangle is bigger than the second one.");
		}
		else if (comparisonOfRectangleArea(r1, r2) == termOfComparison("value1IsSmaller")) {
			System.out.println("The Area of the first rectangle is smaller than the second one.");
		}
		else {
			System.out.println("Two rectangles are same in area.");
		}
	}

	boolean isItEqualRectangle(Rectangle r1, Rectangle r2) {
		if (comparisonOfTwoValues(r1.width, r2.width) == termOfComparison("same")
				&& comparisonOfTwoValues(r1.height, r2.height) == termOfComparison("same")) {
			return true;
		}
		else {
			return false;
		}
	}

	int comparisonOfRectangleArea(Rectangle r1, Rectangle r2) {
		if (comparisonOfTwoValues(calculateAreaOfRectangle(r1),
				calculateAreaOfRectangle(r2)) == termOfComparison("value1IsBigger")) {
			return termOfComparison("value1IsBigger");
		}
		else if (comparisonOfTwoValues(calculateAreaOfRectangle(r1),
				calculateAreaOfRectangle(r2)) == termOfComparison("value1IsSmaller")){
			return termOfComparison("value1IsSmaller");
		}
		else {
			return termOfComparison("same");
		}
	}

	int comparisonOfTwoValues(double value1, double value2) {
		if (value1 > value2){
			return termOfComparison("value1IsBigger");
		}
		else if (value1 < value2){
			return termOfComparison("value1IsSmaller");
		}
		else {
			return termOfComparison("same");
		}
	}

	int termOfComparison(String term) {
		if (term == "value1IsBigger") {
			return 1;
		}
		else if (term == "value1IsSmaller") {
			return -1;
		}
		else {
			return 0;
		}
	}
}


public class TestRectangle {

	public static void main(String[] args) {

		Rectangle r1 = new Rectangle(1.0, 1.0);
		Rectangle r2 = new Rectangle(1.0, 2.0);
		Rectangle r3 = new Rectangle(2.0, 1.0);

		System.out.println("* * * Comparison of the rectangle * * *");
		System.out.println("* rectangle r2 *");
		r2.printRectangleInformation(r2);
		System.out.println("* rectangle r3 *");
		r3.printRectangleInformation(r3);
		System.out.println("* Comparison *");
		r2.compareRectangle(r2, r3);

		System.out.println("\n* * * Information of the rectangle r1 before changed. * * *");
		r1.printRectangleInformation(r1);

		r1.resizeRectangleWithRatio(r1, 2.0);

		System.out.println("\n* * * Information of the rectangle r1 after changed. * * *");
		r1.printRectangleInformation(r1);
	}

}
```

▲ **Fig. 4.** Source Code of the program written in Java

3-101 : Rectangle class
- 4-5 : 인스턴스 변수 선언
- 7-10 : Rectangle(double width, double height) 생성자
- 12-16 : Rectangle 정보를 출력하는 printRectangleInformation method 정의
- 18-21 : Rectangle을 비교하는 method로 연결해주는 compareRectangle method정의
- 23-26 : Rectangle의 크기를 바꿔주는 resizeRectangle method 정의(call by value)
- 28-30 : Rectangle의 넓이를 계산해주는 calculateAreaOfRectangle method 정의
- 32-39 : 같은 Rectangle인지 출력해주는 printIsItEqualRectangle method 정의
- 41-52 : Rectangle의 넓이의 대소관계를 출력해주는 printComparisonRectangleArea method 정의
- 54-62 : 같은 Rectangle인지 비교해주는 isItEqualRectangle method 정의
- 64-76 : Rectangle의 넓이를 비교해주는 comparisonOfRectangleArea method 정의
- 78-88 : 두 값을 비교해주는 comparisionOfTwoValues method 정의
- 90-100 : 대소비교 용어를 바꿔주는 termOfComparison method 정의

104-129 : TestRectangle class
- 106-127 : main method
  - 108-110 : 객체 생성
  - 112-118 : r2와 r3을 비교하는 method 호출
  - 120-126 : r1의 정보를 바꿔서 출력하는 method 호출(call by value)

3.3. Compare the difference between the program written in C and Java

수업시간에 배운 C언어와 Java의 차이점은 아래의 도표로 정리할 수 있다.

C	Java
Low level system 접근 가능	JVM 위에서 실행
call by value / call by reference	call by value
명시적 메모리 관리	자동 garbage collection

▲ **Table. 1.** Difference between C and Java

Table. 1.에 있는 항목 중 **Fig. 1.** 과 **Fig. 2.** 에서 볼 수 있는 것은 call by value, call by reference이다. **C**는 매개변수의 참조 방식이 **call by value**와 **call by reference**로 두가지 방식이 있지만**, Java**는 **call by value**만 존재한다. C의 포인터는 메모리 주소 값으로 직접 조작할 수 있다. 하지만 Java는 포인터가 없다. 즉, 객체에 대한 참조와 배열 참조가 있지만, 메모리 주소에 대한 직접 접근은 허용되지 않는다. C는 포인터에 대한 포인터를 만들 수도 있지만, Java의 참조는 객체에 대한 접근 기능만 제공한다. [1]
Table. 1.에 있는 항목 외에도 **Fig. 1.** 과 **Fig. 2.**를 통해 다른 차이점을 발견할 수 있다.

```
3  struct rectangle {
4     float width;
5     float height;
6  } ;
```

▲ **Fig. 5.** Structure rectangle written in C

```
3 class Rectangle {
4         private float width;
5         private float height;
```

▲ **Fig. 6.** Class Rectangle written in Java

첫 번째로, **C**에서는 **structure**[1]를 사용하지만 **Java**에서는 **class**[2]를 사용한다. 이 둘의 차이점은, structure는 자료의 집합이고 class는 자료와 method의 집합이라는 것이다. 게다가 class는 캡슐화가 가능하고 상속 가능한 객체이다. 따라서 structure는 메모리 자체를 사용하고자 할 때 더 효율적이고, class는 기능적인 측면이 중요할 때 더 효율적이다.

```
24  void resize(struct rectangle *r, float ratio) {
25      r->width *= ratio;
26      r->height *= ratio;
27  }
```

▲ **Fig. 7.** Function resize written in C

```
23          void resizeRectangleWithRatio(Rectangle r1, double ratio) {
24                  r1.width *= ratio;
25                  r1.height *= ratio;
26          }
```

▲ **Fig. 8.** Method resizeRectangleWithRatio written in Java

두 번째로, **C**에서는 함수에 매개변수로 변수 혹은 주솟값이 사용되지만 **Java**에서는 객체가 사용된다.
세 번째로, **C**의 디자인은 절차적이지만 **Java**는 객체와 그 기능들이 중점이 된다. **Fig. 1.**에서는 프로그램이 절차에 중점을 두어 단계적으로 수행되는 디자인 형태를 띄고 있다. 하지만 **Fig. 2.** 에서는 절차에 중심이 아닌, 객체와 그 기능들에 중점을 두어 기능이 연결되는 디자인 형태를 띄고 있다. 즉, Java는 Object-Oriented Programming의 형태를 띄고 있다.

3.4. Advantages/Disadvantages of each style

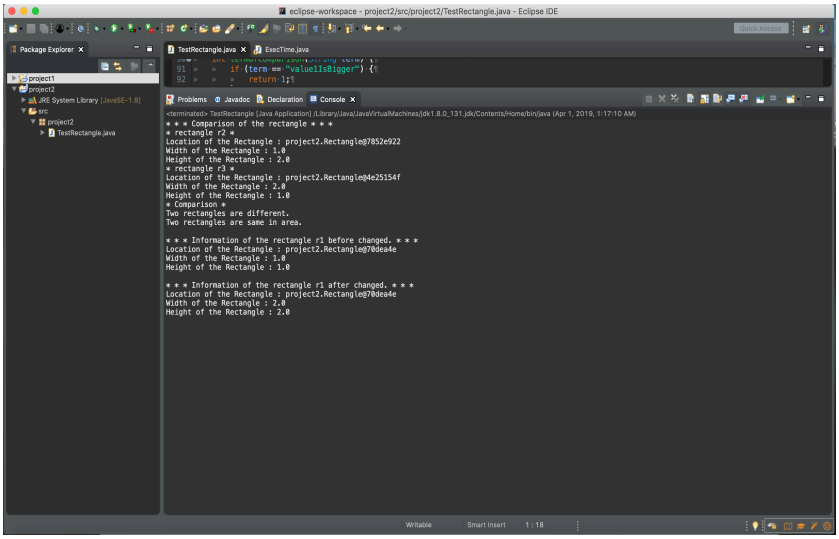
3.3 을 통해 Java 프로그램의 장단점을 도출해 낼 수 있다.
우선, Java 프로그램의 장점은 다음과 같이 정리할 수 있다.
첫 번째, 가독성과 네이밍의 측면에서 **clean code**를 구현하기 쉽다.
두 번째, 코드의 재사용성이 강하다.
세 번째, 유지보수가 쉽다.
네 번째, 대형 프로젝트에 유리하다.
Object-Oriented Programming의 디자인 형태는 객체와 그 기능들이 중심이 되어 code의 가독성이 좋아진다. 특히 Top-down 형태를 띈다면 이는 더욱 극대화될 것이다. 따라서 재사용성이 강하고 유지보수가 쉽다. 이는 대형 프로젝트를 진행함에 있어서 큰 이점이 된다. 대형 프로젝트는 code를 개발하는 개발자의 수도 많고 code의 줄 수도 길다. 이는 가독성이 필수적인 요구사항이 된다.
다음으로, Java 프로그램의 단점은 다음과 같이 정리할 수 있다.
첫 번째, 디자인 능력이 요구된다.
두 번째, **code**의 줄 수가 증가한다.
이는 **Fig. 1.** 와 **Fig. 2.** 를 비교하여 알 수 있다. **Fig. 1.** 은 생각의 흐름을 그대로 적었지만, **Fig. 2.** 는 기능에 따라 잘 설계된 code이다. 따라서 이는 설계를 잘 할 수 있는 디자인 능력이 요구된다. 게다가 설계된 기능에 따라 만들어진 code는 줄 수가 길어질 수 밖에 없다. 이는 **Fig. 2.**에서도 확인할 수 있다.

---

[1] structure : 하나 이상의 변수를 묶어 그룹화하는 사용자 정의 자료형이다. 다른 자료형도 그룹화 할 수 있다.

[2] class : 객체를 만들어내기 위한 틀이다.

4. Program Results



▲ **Fig. 5.** Result of the project written in Java

5. Conclusion

이 프로젝트에서 고찰할 점은 'Java 프로그램과 C 프로그램의 스타일이 어떻게 다른가' 이다.
이에 대한 프로젝트의 결론은 다음과 같다.

1. **C**는 매개변수의 참조 방식이 **call by value**와 **call by reference**로 두가지 방식이 있지만, **Java**는 **call by value**만 존재한다.
2. **C**에서는 **structure**를 사용하지만 **Java**에서는 **class**를 사용한다.
3. **C**의 디자인은 절차적이지만 **Java**는 객체와 그 기능들이 중점이 된다.

이러한 결론을 통해 얻은 Java 프로그램의 장점은 다음과 같다.

1. 가독성과 네이밍의 측면에서 **clean code**를 구현하기 쉽다.
2. 코드의 재사용성이 강하다.
3. 유지보수가 쉽다.
4. 대형 프로젝트에 유리하다.

따라서, Java의 장점을 살리는 프로그래밍 방식은 Object-Oriented Programming이다. 즉, 객체지향적 디자인 설계가 **Java**의 장점을 극대화하는 방안이다. **Java** 프로그래밍을 하기 전에 객체 디자인 설계를 통해 기능 중심의 구현을 해야 한다. 이때 모든 언어의 프로그래밍에서 중요한 디자인은 **Top-down design**이다. **Top-down design**을 함으로써 **clean-code**의 구현을 극대화하여 프로젝트에 이점을 줄 것이다.

6. References

[1] 위키백과, 우리 모두의 백과사전, "자바와 C++의 비교", https://ko.wikipedia.org/wiki/%EC%9E%90%EB%B0%94%EC%99%80_C%2B%2B%EC%9D%98_%EB%B9%84%EA%B5%90

7. Evaluation

평가 항목	학생 자체 평가 (리포트 해당 부분 표시 및 간단한 의견)	평가 (빈칸)	점수 (빈칸)
Java Style - 동작? - 요구 사항 만족?	Java Style로 프로그램 구현 - 프로그램이 동작하며 동작 사진을 첨부함 - class Rectangle을 별도로 만들어 모든 관련 메소드(함수)들을 그 안에 포함함 - 프로그램의 테스트 동작은 별도의 class TestRectangle을 만들어서 수행함		
두 방법 비교	- 두 방법을 비교함 - 사진을 첨부하여 비교함		
기타	프로젝트의 목적에 따른 고찰점을 제시하여 결론을 정리함		
총평/계	평가 항목을 모두 만족하였으며, 직접 프로그래밍하였음을 사진을 통해 입증함		