2019. 3. 28.

소프트웨어프로젝트1

Project 1-1

최강임 교수님

컴퓨터정보공학부 2015722087

김민철

P1.

Introduction

이번 문제는 주어진 배열, 혹은 임의의 배열을 내림차순(큰 수 -> 작은 수)으로 정렬하고 출력하는 문제이다.

Result

```
<terminated> p1_1_1 [Java Application]
Array Before Bubble Sort
5 90 35 45 150 3
Array After Bubble Sort
150 90 45 35 5 3
```

정렬되기 이전의 배열은 과제 문제에서 사용한 배열 그대로를 사용하였다. Int형 변수 tmp를 선언하여 swap시 데이터를 잠시 저장하는 용도로 사용하였고, 배열의 값들을 현재 값과 현재 값의 뒤의 있는 값을 비교하여 더 작은 수가 앞에 있을 경우에는 정렬이 필요하기 때문에 조건문을 통하여 swap을 해주었다. 이를 반복하여 최종적으로 내림차순으로 정렬한 결과를 출력했다.

Code

```
public class p1_1_1 {

	public static void main(String[] args)
	{
		int array[] = {5, 90, 35, 45, 150, 3}; // 정렬할 배열
		int tmp; // swap 할때 사용할 변수

		System.out.println("Array Before Bubble Sort");
		for(int i=0; i<array.length; i++) // 화면에 배열 출력.
		{
			System.out.print(array[i] + " ");
		}
		System.out.print("\n");

		for(int i=0; i<array.length-1; i++) // 배열의 길이 만큼 cycle.
		{
			for(int j=0; j<array.length-1; j++) // 1 cycle. 반복
			{
				if(array[j] < array[j+1]) // 정렬이 필요한가?
				{
					// SWAP.
					tmp=array[j];
					array[j]=array[j+1];
					array[j+1]=tmp;
				}
			}
		}


		System.out.println("Array After Bubble Sort");
		for(int i=0; i<array.length; i++) // 화면에 배열 출력.
		{
```

```
                    System.out.print(array[i] + " ");
                }

        }
}
```

Consideration

이번 문제는 자바를 공부하고 처음으로 받은 문제이자, 처음으로 자바로 코딩을 해보게 된 문제이다. 지금 보면 간단한 문제이지만, 문제를 읽고, 알고리즘을 어느 정도 생각하고 이제 작성을 시작하려는 순간, 막상 어떻게 해야 될지 몰라서 자바 책을 계속 들여다보면서 작성을 했었다. 배열과 입출력이 기존에 해왔던 언어들과 달라서 책을 보고 이해하면서 하느라 오래 걸려서 답답하기도 했지만 다음 문제부터는 조금 적응이 된 상태에서 해볼 수 있을 것 같다.

P2.

Introduction

이번 문제는 주어진 문자열을 공백을 기준으로 나누고, 특수 문자를 제외한 알파벳 만을 출력하는 문제이다.

Result

```
p1_1_2 [Java Application] C:₩Program File
USER INPUT :
i have2&* n()*o ide%&57a.
OUTPUT :
i
have
no
idea
```

이 문제에서도 사용자 입력은 과제 문제에 있던 입력 값 그대로 사용하였다. 스캐너 클래스의 next() 메소드를 사용하면 공백을 기준으로 끊은 문자열을 받기 때문에 next() 메소드를 이용하여 공백으로 문자열을 구분하여 다루었고, 알파벳 만을 출력하도록 하기 위해서 단어의 문자를 모두 알파벳인지 확인하기 위해 스캐너 클래스의 charAt(int a) 메소드를 이용하여 문자 하나씩을 다루어서 아스키 코드의 알파벳 값과 비교하여 확인하였다. 이때 인자(int a)는 문자열의 몇 번째를 가리키는지 나타내는 index 값이다. 확인된 알파벳을 출력하고, 한 단어의 출력이 끝났으면 다음 단어를 출력하고, 문자열이 모두 처리되면 프로그램이 종료된다.

Code

```
import java.util.Scanner;
//i have2&* n()*o ide%&57a.

public class p1_1_2 {

	public static void main(String[] args)
	{
		Scanner scanner = new Scanner(System.in);

		String tmp; // 단어를 입력받을 스트링.

		System.out.println("USER INPUT : ");
		tmp=scanner.next(); // 문자열 입력받기
		System.out.println("OUTPUT :");

		while(true) // 문자열이 단어 몇개인지 모르기때문에 무한루프로 반복
		{
			for(int i=0; i<tmp.length(); i++)
			{
				// tmp 배열에 들어있는 단어의 각 문자 하나하나가 알파벳일때만 출력
				if( ( (int)tmp.charAt(i)>=65 && (int)tmp.charAt(i)<=90 ) ||
( (int)tmp.charAt(i)>=97 && (int)tmp.charAt(i)<=122))
```

```
                                System.out.print(tmp.charAt(i));
                        }
                        // 다음 단어를 tmp 배열에 불러오기.
                        System.out.println("");
                        tmp=scanner.next();

                        if(tmp == null) // 문자열이 종료되었으면 반복문 탈출
                                break;
                }

                scanner.close(); // 스캐너 종료
        }
}
```

Consideration

이번 문제부터는 적응이 될 것 같았지만 스캐너를 이해하는데 시간이 조금 걸렸다. C언어에서 했던 문자의 입력을 받는 것과 문자열을 다루는 것이 자바에서는 달라서 공부하느라 꽤 오래 걸렸다. 공부를 하고 막상 코드를 작성해보니 자바가 조금 더 쉽게 구현할 수 있도록 되어있다는 느낌을 받았고, 아직은 잘 모르겠지만 왜 자바가 많이 사용되어 점유율이 높은 언어인지 조금 알 것 같다. 스캐너 클래스는 앞으로도 자주 이용할 것 같고, 중요할 것 같아 공부를 더 해야 될 것 같다.

P3.

Introduction

이번 문제는 1~6 사이의 숫자가 랜덤으로 나오는 주사위를 2개 구현하여 그 두 주사위의 값들을 더한 값, 즉 두 주사위 값의 합을 출력하는 문제이다.

Result

```
<terminated> p1_1_3 [Java Application] C:₩P
The first die comes up 3
The second die comes up 4
Your total roll is  7
```

Math 클래스의 random() 메소드를 이용하여 0~1 사이의 랜덤 실수 값을 받아서, 거기에 6을 곱하면 0.xx, 1.xx, ... ,5.xx 가 되고, 거기에 1을 더한 다음, 이 값을 정수형으로 강제 형 변환해주어서 1~6 사이의 랜덤 값이 나오게 하였고, 첫번째 주사위, 두번째 주사위의 랜덤 값을 출력하고, 마지막에 두 주사위의 값들의 합을 구해 출력하였다.

Code

```
public class p1_1_3 {

	public static void main(String[] args)
	{
		// Math.random() 메소드는 0~1 사이의 값이 랜덤으로 나온다 1~6이
나오게 하기위해 *6+1
		int dice1 = (int)(Math.random() * 6 + 1); // 첫번째 주사위
		int dice2 = (int)(Math.random() * 6 + 1); // 두번째 주사위

		// 결과값 출력
		System.out.println("The first die comes up " + dice1);
		System.out.println("The second die comes up " + dice2);
		System.out.println("Your total roll is  " + (dice1 + dice2));
	}
}
```

Consideration

0~1사이의 실수 값을 반환하는 메소드인 Math.random() 를 참고하라고 과제에서 주어졌을 때, 그냥 주어진 대로 코드를 짜고 실행했을 때 문제는 없었지만, 어떤 자료 형이길래 int 형으로 자료 형을 변환할까. 왜 6을 곱하고 1을 더할까 이해가 되지 않아서 구글에 검색해서 알아 보았다. Math 클래스의 메소드인 random() 은 double 형의 자료를 반환하기 때문에 정수형인 int 형으로 강제 형 변환을 해준 것이다. 간단한 문제이지만 Math 라는 클래스와 강제 형 변환까지 공부할 수 있는 문제였다.

P4.

Introduction

이 문제는 0~20 사이의 랜덤 값 10개를 오름차순으로 정렬하여 출력하고, 거기에 숫자를 삽입하고, 삭제하고, 탐색하는 명령을 입력받아 수행하는 프로그램을 구현한다. 숫자는 모든 명령이 종료 되고 나서 정렬된다. (메인 클래스가 아닌 새로운 클래스를 구현하고 사용해야하고, 숫자는 중복되지 않는다. 또한 정렬 API를 사용하면 안된다.)

Result

```
p1_1_4 [Java Application] C:₩Program Files₩Java₩jdk-11.0.
List: 2 3 8 10 11 12 13 16 18 20
Input your command: a
Input number to add: 9
List: 2 3 8 9 10 11 12 13 16 18 20
Input your command: r
Input number to remove: 2
List: 3 8 9 10 11 12 13 16 18 20
Input your command: r
Input number to remove: 20
List: 3 8 9 10 11 12 13 16 18
Input your command: s
Input number to search: 10
Index of 10 is 3
List: 3 8 9 10 11 12 13 16 18
Input your command:
```

ArrayList를 사용했기 때문에 구현해야 할 메소드는 배열의 초기 랜덤 값 10개를 입력하는 것과, 오름차순 정렬을 하는 메소드이고, 이 두개의 메소드는 Oper 클래스를 만들어 구현하였다. Oper 클래스는 생성자에서 ArrayList를 받아와서 Oper 클래스 메소드에서 메인 함수에서 선언된 ArrayList를 다룰 수 있게 하였다. 배열의 초기값을 입력하는 메소드는 인서트 할 때마다 랜덤 값이 배열에 존재하는 값인지 확인하고 인서트하여 중복이 생기지 않게 했고, 정렬 메소드는 버블 정렬을 구현하였다. 이외에는 ArrayList의 메소드를 활용하여 문제를 해결하였다. 명령어에 a(insert),r(remove),s(search)를 입력할 수 있고, e를 입력하면 프로그램이 종료되게 만들었다. 이외의 명령어를 입력하면 오류 메시지를 출력하게 하였다.

Code

```
import java.util.Scanner;
import java.util.ArrayList;

public class p1_1_4 {

	public static void main(String[] args)
	{
		Scanner scanner = new Scanner(System.in);
		char cmd; // 명령어. a, r, s, e
		int a; // 입력받을 정수
```

```
                ArrayList<Integer> array = new ArrayList(); // ArrayList 객체 생성

                Oper op = new Oper(array); // 초기 배열 생성과 정렬 역할 수행
                op.birth(); // 초기 배열 생성

                while(true) // 게임 시작
                {
                        op.my_sort(); // 배열 정렬

                        // 현재 리스트 출력한다.
                        System.out.print("List: ");
                        for(int i=0; i<array.size(); i++)
                                System.out.print(array.get(i)+" ");

                        // 명령 입력
                        System.out.print("\nInput your command: ");
                        cmd = scanner.next().charAt(0);

                        // 명령에 따른 작동
                        if(cmd == 'a') // 추가
                        {
                                System.out.print("Input number to add: ");
                                a = scanner.nextInt();

                                if(array.indexOf(a) == -1) // 중복 방지.
                                        array.add(a);
                        }
                        else if(cmd == 'r') // 삭제
                        {
                                System.out.print("Input number to remove: ");
                                a = scanner.nextInt();

                                array.remove((Integer)a); // a라는 값을 찾아서 제거
                        }
                        else if(cmd == 's') // 검색
                        {
                                System.out.print("Input number to search: ");
                                a = scanner.nextInt();

                                if(array.indexOf(a) != -1) // 존재할 경우
                                        System.out.println("Index of " +a+ " is "
+array.indexOf(a)); // a의 인덱스 출력
                                else // 존재하지 않을 경우
                                        System.out.println("Index of "+a+" is not exist.");
                        }
                        else if(cmd == 'e') // 종료
                        {
                                break;
                        }
                        else // 잘못된 명령
                                System.out.println("Error : that command is non-exist");
                }

                scanner.close();
                return;
        }
}

class Oper
{
        ArrayList<Integer> ar;
        Oper(ArrayList<Integer> arr) {ar = arr;};

        void birth() // 초기 배열 생성
```

```
	{
		int rand;

		for(int i=0; i<10; i++) // 랜덤 값 10개 입력
		{
			rand = (int)(Math.random()*20+1);
			for(int j=0; j<10; j++)
			{
				if(ar.size() == j) // 배열의 끝이면 랜덤값 입력
				{
					ar.add(rand); // 랜덤값 입력
					break;
				}
				else if(rand == ar.get(j)) // 중복 값 확인 0번째칸 ~ 끝까지
				{
					i--;
					break; // 중복이면 랜덤값 재생성
				}
			}
		}
	}

	void my_sort() // 정렬하는 메소드
	{
		int tmp1;
		int tmp2;

		// 버블 정렬
		for(int i=0; i<ar.size()-1; i++)
		{
			for(int j=0; j<ar.size()-1; j++)
			{
				if(ar.get(j) > ar.get(j+1)) // 정렬이 필요한가?
				{
					// SWAP.
					tmp1=ar.get(j);
					tmp2=ar.get(j+1);

					ar.remove(j);
					ar.remove(j); // 위에서 j번째 칸이 지워졌기 때문에 j+1
번째가 j가되었다.

					ar.add(j, tmp2);
					ar.add(j+1, tmp1);
				}
			}
		}
	}
}
```

Consideration

처음에 ArrayList 라는 게 내가 구현해야되는 클래스 인줄 알고, 한참을 걸려 삽입, 삭제, 탐색 기능이 있는 ArrayList를 만들어 과제를 진행하던 중, 문제를 다시 읽어보다가 뭔가 이상해서 구글에 ArrayList를 검색하니 원래 존재하는 클래스 였다는 걸 알고 조금 허탈한 마음으로 문제를 마무리했다. 오히려 이미 존재하는 클래스를 사용하려니 메소드가 무엇이 있는지, 어떤 값을 반환하고 어떤 동작을 하는지 잘 몰라 ArrayList 클래스를 공부하고 진행해야했다. 이 문제를 해결하면서 자바의 여러가지 제공되는 클래스를 알면 많이 유용하겠다는 생각이 들었다.

P5.

Introduction

이 문제는 Boggle 이라는 연속으로 이어진 알파벳으로 이루어진 단어들을 NxN 크기의 퍼즐에서 찾는 간단한 게임이다.

1. 3x3 크기의 퍼즐을 만들기.
2. 퍼즐의 알파벳은 랜덤으로 할당.
3. 사용자가 단어를 입력하면 단어를 퍼즐에서 찾는다.
4. 단어는 최소한 3글자보다 길어야 한다.

(메인 클래스가 아닌 새로운 클래스를 생성하여 사용하고, 입력한 단어가 3글자보다 작을 경우 오류 메시지를 출력하라.)

Result

```
-------------
| F | Z | O |
-------------
| Z | B | X |
-------------
| H | M | W |
-------------
Input word: FZX
FZX
Input word: FZM
Non-exist
Input word: FF
ERROR : The input word size is less than 3
Input word:
```

우선 퍼즐을 코드에서도 알아보기 쉽게 하기 위하여 그림과 같이 3x3 의 2차원 배열을 이용하여 배열을 생성하였다. 클래스 Boggle 을 만들어 모든 동작을 Boggle 클래스의 메소드를 이용하여 작동하게 하였고, 메인 클래스는 최대한 적은 코드로 구현했다. 단어를 입력 받고, 퍼즐에서 찾는 과정은 알파벳 하나 씩 찾고, 찾은 알파벳이 이전 알파벳(단어에서 찾은 알파벳의 앞에 있는 알파벳)과 같은 위치는 아닌지 확인하고, 인접한 알파벳인지 확인할 때는 알파벳 위치의 거리(2차원 배열이기 때문에 1칸.)가 가로, 세로 모두 1칸 이내인지 확인하기 위해 Math 클래스의 abs 메소드(절대값으로 변환)를 이용하였다.(가로, 세로의 떨어진 거리가 각각 1이어야 한칸이다. 가로 1, 세로 1 차이는 대각선 칸에 있는 것이고, 가로 1차이는 옆 칸에 있는 것이고, 세로 1차이는 위 또는 아래 칸에

있는 것이다.) 이전 알파벳과 연속된 경로인 경우 현재 몇 번째 알파벳을 탐색하는지 나타내는 int형의 cnt 값을 +1 해서 해당 알파벳을 찾았다는 표시를 해주고, 현재 cnt값이 입력한 단어의 길이와 같은 것은 단어의 마지막 알파벳을 탐색하였다는 의미이기 때문에 단어를 모두 찾았다는 것이고, 같지 않은 경우 아직 단어의 알파벳을 모두 찾았다는 뜻이 아니다. 만약 3x3 퍼즐을 모두 다니면서 알파벳을 찾는 동안 입력한 단어의 모든 알파벳을 찾지 못한 경우는 단어가 없는 경우이므로 단어가 없다는 메시지를 출력하였다. 무한 루프를 탈출하기위해 명령어에 $EXIT를 입력하면 프로그램이 종료될 수 있게 하였다.

Code

```
import java.util.Scanner;

public class p1_1_5 {

	public static void main(String[] args)
	{
		Boggle game = new Boggle(); // 퍼즐이 진행될 클래스 객체 생성
		game.play_game(); // 게임 시작


		Scanner scanner = new Scanner(System.in);
		String word;


		while(true)
		{
			System.out.print("Input word: ");
			word = scanner.next(); // 찾을 단어 입력

			if(game.input(word) == false) // 게임 실행 -> 종료 여부 판단
			{
				System.out.println("Program END");
				break;
			}
		}
	}
}

class Boggle
{
	char[][] boggle = new char[3][3]; // 퍼즐이 될 배열

	void play_game()
	{
		// 3x3 크기의 랜덤 알파벳 배열 생성
		for(int j=0; j<3; j++)
		{
			for(int i=0; i<3; i++)
				boggle[j][i]=(char)(Math.random()*26+65);
		}

		System.out.println("-------------");
		System.out.println("| "+boggle[0][0]+" | "+boggle[0][1]+" | "+boggle[0][2]+"
|");
		System.out.println("-------------");
		System.out.println("| "+boggle[1][0]+" | "+boggle[1][1]+" | "+boggle[1][2]+"
|");
		System.out.println("-------------");
		System.out.println("| "+boggle[2][0]+" | "+boggle[2][1]+" | "+boggle[2][2]+"
|");
```

```
                System.out.println("-------------");
        }

        boolean input(String word)
        {
                if(word.equals("$EXIT")) // 프로그램 강제종료
                        return false;

                if(word.length() < 3) // 글자수가 3개보다 작으면 오류.
                {
                        System.out.println("ERROR : The input word size is less than 3");
                        return true;
                }

                int cnt=0; // 몇번째 알파벳이 진행중인지 표시. 첫알파벳 = 0
                int[][] pos = new int [word.length()][2]; // [알파벳 구분(첫번째 알파벳 =
0)][퍼즐에서의 위치 x,y]

                for(int i=0; i<word.length(); i++) // 초기값 -10로 초기화
                {
                        pos[i][0] = -10;
                        pos[i][1] = -10;
                }

                for(int len=0; len<9; len++) // 단어찾기
                {
                        for(int a=0; a<3; a++) // 알파벳 찾기 행 구분.
                        {
                                for(int b=0; b<3; b++) // 알파벳 찾기 열 구분.
                                {
                                        if(word.charAt(cnt) == boggle[a][b]) // 알파벳을
찾았다면,
                                        {
                                                for(int i=0; i<word.length(); i++) // 찾은
알파벳이 적절한지 확인
                                                {
                                                        if(pos[i][0] == a && pos[i][1] ==
b) // 중복된 위치다.
                                                                break;
                                                        else if(cnt == 0 || ( Math.abs(a-
pos[cnt-1][0]) <= 1 && Math.abs(b-pos[cnt-1][1]) <= 1 )) // 첫 알파벳이거나, 이전 알파벳과 인접한
위치인가 ?
                                                        {
                                                                pos[cnt][0]=a;
                                                                pos[cnt++][1]=b;
                                                                break;
                                                        }
                                                }
                                        }
                                        if(cnt==word.length()) // 다찾았다면,
                                                break;
                                }
                                if(cnt==word.length()) // 다찾았다면,
                                        break;
                        }

                        if(cnt == word.length()) // 단어를 찾았다면,
                        {
                                System.out.println(word);
                                break;
                        }
                        else if(len == 8) // 단어를 못찾았다면,
                                System.out.println("Non-exist");
                }
                return true;
```

```
            }
}
```

Consideration

이 프로그램을 구현하면서 가장 어려웠던 순간은 알고리즘을 생각하는 과정에서 단어를 어떻게 찾을 것인지 고민할 때가 가장 어려웠다. 첫 알파벳을 찾고, 다음 알파벳을 찾는 과정을 어떻게 해야 될지 여러가지 방법을 시도해보면서 고민해보았다. 처음에는 이전 알파벳의 인접한 위치(좌,우,상,하,대각선)을 확인해보는 방법을 시도해보았는데 가장 왼쪽에 있는 알파벳은 왼쪽에 데이터가 없기 때문에 예외 처리를 해야 되는데 그렇게 되면 코드가 엄청나게 길어질 것 같다는 생각이 들어 포기하고 다른 방법을 생각하던 중, 일단 알파벳이 퍼즐에 있는지 찾고, 그 알파벳이 이전 알파벳과 인접 한지 확인을 해보는 게 가장 좋겠다는 생각이 들어 그 방법으로 구현하였다. 만약 이렇게 알고리즘을 바꾸지 않고 기존의 방법으로 끝까지 코드를 작성했다면 지금보다 훨씬 복잡하고 성능이 좋지 않은 프로그램이 나왔겠지만, 고민을 하여 더 좋은 프로그램을 구현했다는 것에 뿌듯한 문제였다.

P6.

Introduction

이 문제는 클래스의 상속을 이용하여 자동차를 설계하는 프로그램을 구현하는 것이다. 클래스의 이름과 인스턴스 변수, 메소드를 출력하라. Brand 와 Model 이라는 슈퍼 클래스를 이용하고 최소한 6개의 자동차 클래스를 만들어라.(클래스의 상속을 문서에 그려라)

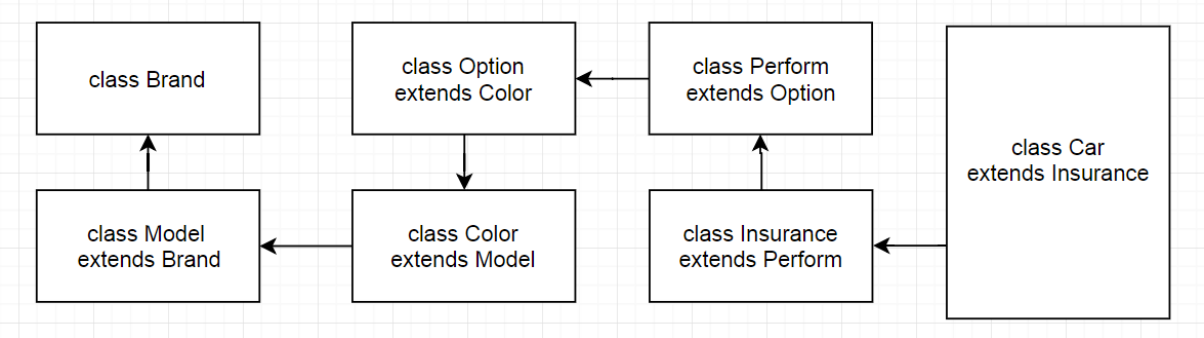
Result

```
What you want to Brand ?
KIA
What you want to Type ?
Sedan
What you want to Color ?
Black
What you want to Option ?
Full_Option
What you want to Performance ?
Best_Performance
What you want to Insurance ?
10years_Warranty

This is My Car !!
Brand : KIA
Type : Sedan
Color : Black
Option : Full_Option
Performance : Best_Performance
Insurance : 10years_Warranty
```

우선 Brand와 Model을 부모 유전자로 사용하고 클래스를 6개 구현하여 자동차 설계 프로그램을 구현하는 것을 중점으로 구현하였다. 별다른 조건이 없어 최대한 단순한 프로그램을 설계하였고, 차를 결정하는데 클래스를 6가지를 정했다.(브랜드, 타입, 색상, 옵션, 성능, 보험) 그리고 최종적으로 Car 클래스에서 모든 클래스의 멤버에 접근 할 수 있게 하여, 결론적으로는 Car 클래스의 객체가 접근 할 수 있는 멤버가 6개가 되었고, 이 6개의 멤버는 각각 다른 클래스에 선언 되어 있다. 결국 보다시피 Car 클래스 객체만을 이용하여 모든 정보를 입력하고 출력하였다.

Hierarchy



Car 클래스는 최종적으로 클래스 6개(Brand, Model, Color, Option, Perform, Insurance)의 멤버에 접근 할 수 있기 때문에 클래스 내부에 선언된 멤버 변수가 없어도 데이터를 입력하고 출력 할 수 있다.

Code

```
import java.util.Scanner;
public class p1_1_6 {

	public static void main(String[] args)
	{
		String select; // 입력받은 데이터를 임시로 저장할 변수
		Scanner scan = new Scanner(System.in);

		Car car = new Car(); // Car 객체 생성

		System.out.println("What you want to Brand ?");
		select = scan.next();
		car.brand = select; // Car 객체의 brand 변수에 select 입력

		System.out.println("What you want to Type ?");
		select = scan.next();
		car.model = select; // Car 객체의 model 변수에 select 입력

		System.out.println("What you want to Color ?");
		select = scan.next();
		car.color = select; // Car 객체의 color 변수에 select 입력

		System.out.println("What you want to Option ?");
		select = scan.next();
		car.option = select; // Car 객체의 option 변수에 select 입력

		System.out.println("What you want to Performance ?");
		select = scan.next();
		car.perform = select; // Car 객체의 perform 변수에 select 입력

		System.out.println("What you want to Insurance ?");
		select = scan.next();
		car.ins = select; // Car 객체의 ins 변수에 select 입력

		car.show(); // car 객체의 정보 출력

		scan.close();
```

```
                return;
        }
}
class Car extends Insurance // 6개의 클래스의 정보를 모두 상속받음
{
        void show() {
                // Car 클래스 내에 선언되지 않았지만 상속받은 멤버 변수를 출력
                System.out.println("\nThis is My Car !!");
                System.out.println("Brand : "+this.brand);
                System.out.println("Type : "+this.model);
                System.out.println("Color : "+this.color);
                System.out.println("Option : "+this.option);
                System.out.println("Performance : "+this.perform);
                System.out.println("Insurance : "+this.ins);
        }
}

class Brand
{
        String brand; // default : 같은 패키지라면 접근 가능
}

class Model extends Brand
{
        String model; // default : 같은 패키지라면 접근 가능
}

class Color extends Model
{
        String color; // default : 같은 패키지라면 접근 가능
}

class Option extends Color
{
        String option; // default : 같은 패키지라면 접근 가능
}

class Perform extends Option
{
        String perform; // default : 같은 패키지라면 접근 가능
}

class Insurance extends Perform
{
        String ins; // default : 같은 패키지라면 접근 가능
}
```

Consideration

클래스의 상속을 이용하여 자동차를 설계해야 하고, Brand와 Model을 부모 클래스로 사용해야한다는 조건에서 어떻게 해야 저 두 클래스를 부모 클래스로하여 클래스를 6개 구현하고 자동차를 설계하는 프로그램을 구현할 수 있을까라는 생각을 하면서 오히려 주어진 조건이 없어서 좀 더 막막했던 문제였다. 이전 문제들은 구체적인 조건과 보기가 있어 그대로 만들었지만, 이번에는 비교적 자유도가 높은 문제였다. 상속이 가장 중요한 핵심인 것 같아 상속을 공부하고 구현하였으며, 처음에는 클래스가 여러 개를 상속받을 수 는 없나? 하고 찾던 중 interface, implements를 알게 되었으나, 아직 사용하기에는 미숙하여 상속을 여러 번 하는 걸로 진행하였다. 별거 아닌 듯 하지만, 상속이라는 기능이 나중에 복잡한 프로그램을 구현할 때 굉장히 유용할 것 같다는 생각이 들었다.