REPORT 1

소프트웨어프로젝트 04 분반

담 당 교 수 : 이남규 교수님
학 과 : 영어영문학과
학 번 : 20151816
이 름 : 차유진
제 출 일 : 2020.04.06

1. 종합소득 세율 계산

A. 문제

아래 종합소득 세율표에 있는 2020 년 귀속 종합소득세율표를 참고하여 소득 금액별 세부담(소득세, 지방소득세)를 계산하시오.

- 사용자가 소득 금액을 입력하면, 소득세와 지방소득세를 계산하여 출력한다.
- 지방세는 소득세의 10%이다.

종합소득세율		
과세표준	세율	누진공제액
1,200만원 이하	6%	-
1,200만원 초과~4,600만원 이하	15%	108만원
4,600만원 초과~8,800만원 이하	24%	522만원
8,800만원 초과~1억 5,000만원 이하	35%	1천490만원
1억 5,000만원 초과~3억원 이하	38%	1천940만원
3억원 초과~5억원 이하	40%	2천540만원
5억원 초과	42%	3천540만원

B. 해결 방법

1. 소득 금액을 나타내는 변수 'income', 소득세를 나타내는 변수 'income_tax'와 지방소득세를 나타내는 변수 'region_tax'를 선언한다.
2. 소득 금액을 입력 받아 'income'에 저장한다.
3. 'income'이 0 이상의 정수인지 확인한다.
   아닐 경우, "입력한 금액이 잘못됐습니다." 메세지 표시 후 프로그램을 종료한다.
4. if 조건문을 사용해 'income'이 종합소득세율표 중 어느 범위에 해당하는지 파악한다.
5. 해당하는 범위에 따라 정해진 세율을 'income'에 곱한 값이 'income_tax'가 된다.
6. 'income_tax'에 0.1 (10%)를 곱한 값이 'region_tax'가 된다.
7. 'income_tax'와 'region_tax' 결과를 출력한다.

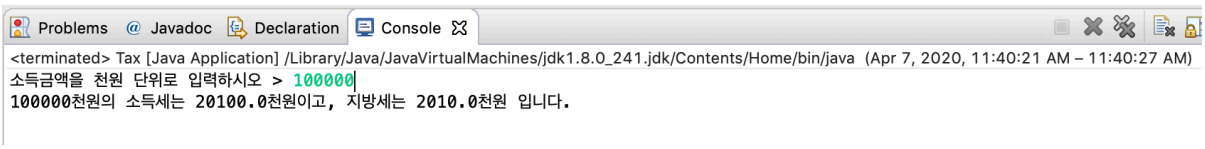
C. 특이사항

- 'income_tax'와 'region_tax'에는 소수점자리를 곱해 주어야 하므로 double 형을 사용하였다.

D. 소스코드 및 주석

```
/*
 파일명: Tax.java
 입력: 소득 금액
 출력: 소득세 및 지방소득세
 */

import java.util.Scanner;
public class Tax {
	public static void main(String[] args) {
		int income; // 입력받을 소득금액 (천원 단위로 입력받기 때문에 int형으로)
		double income_tax; // 출력할 소득세 (밑에서 소수와 곱해야 하기 때문에 double형으로)
		double region_tax; // 출력할 지방소득세

		Scanner scanner = new Scanner(System.in);
		System.out.print("소득금액을 천원 단위로 입력하시오 > ");
		income = scanner.nextInt(); // income 변수에 소득금액 입력받기

		if(income>=0) { // 소득 입력 받을 때 0 이상이 되어야 함. 0보다 크거나 같은 정수가 아닐 경우 else로 넘어감
			if(income<=12000) { // 소득금액이 12000천원(1천2백만원) 이하일 경우
				income_tax = income*0.06; // 소득세 = 소득금액의 6%
				region_tax = income_tax*0.1; // 지방소득세 = 소득세의 10%
				}
			else if(income>12000 && income<=46000) { // 소득금액이 1천2백만원 초과, 4천6백만원 이하일 경우
				income_tax = income*0.15 - 1080; // 소득세 = 소득금액의 15%
				region_tax = income_tax*0.1; // 지방소득세 = 소득세의 10%
				}
			else if(income>46000 && income<=88000) { // 소득금액이 4천6백만원 초과, 8천8백만원 이하일 경우
				income_tax = income*0.24 - 5220; // 소득세 = 소득금액의 24%
				region_tax = income_tax*0.1; // 지방소득세 = 소득세의 10%
				}
			else if(income>88000 && income<=150000) { // 소득금액이 8천8백만원 초과, 1억5천만원 이하일 경우
				income_tax = income*0.35 - 14900; // 소득세 = 소득금액의 35%
				region_tax = income_tax*0.1; // 지방소득세 = 소득세의 10%
				}
			else if(income>150000 && income<=300000) { // 소득금액이 1억5천만원 초과, 3억원 이하일 경우
				income_tax = income*0.38 - 19400; // 소득세 = 소득금액의 38%
				region_tax = income_tax*0.1; // 지방소득세 = 소득세의 10%
				}
			else if(income>300000 && income<=500000) { // 소득금액이 3억원 초과, 5억원 이하일 경우
				income_tax = income*0.40 - 25400; // 소득세 = 소득금액의 40%
				region_tax = income_tax*0.1; // 지방소득세 = 소득세의 10%
				}
			else {  // 소득금액이 5억원 초과일 경우
				income_tax = income*0.42 - 35400; // 소득세 = 소득금액의 42%
				region_tax = income_tax*0.1; // 지방소득세 = 소득세의 10%
				}
			System.out.print(income + "천원의 소득세는 " + income_tax + "천원이고, 지방세는 " + region_tax + "천원 입니다.");
			// 입력받은 소득금액에 대한 소득세와 지방소득세를 출력
		}
		else System.out.print("입력한 금액이 잘못됐습니다."); // 위에서 입력받은 숫자가 0보다 크거나 같은 정수가 아닐 경우 출력
		scanner.close(); // 스캐너 닫아주기
	}
}
```

E. 입력 및 출력 결과 예시



2. 60 갑자

A. 문제

년도를 입력 받아 60 갑자로 무슨 해인지 출력하는 프로그램을 쓰시오.

B. 해결 방법

1. 십간과 십이지에 대한 1 차원 배열 'gan'과 'ji'를 각각 생성한다.
2. 입력받을 년도를 나타내는 변수 'year'과 임시변수 i, j 를 선언한다.
3. 년도를 입력 받아 'year'에 저장한다.
4. 십간은 10 년마다 되풀이되고, 십이지는 12 년마다 되풀이 되기 때문에, 'year'을 10 과 12 로 나누어 나온 나머지 수를 각각 임시 변수(i, j)에 저장한다.
5. 임시 변수를 활용해 십간 배열의 i 번째 데이터와 십이지 배열의 j 번째 배열을 출력한다.

C. 특이사항

- 십간은 갑(甲), 을(乙), 병(丙), 정(丁), 무(戊), 기(己), 경(庚), 신(辛), 임(壬), 계(癸),
  십이지는 자(子), 축(丑), 인(寅), 묘(卯), 진(辰), 사(巳), 오(午), 미(未), 신(申), 유(酉), 술(戌), 해(亥)
  순으로 통용되지만, 60 갑자 계산법에 따라 0 년은 '경신'년이므로 배열의 순서를 달리해주었다.
- 나머지 수에 따른 십간과 십이지는 다음과 같다.

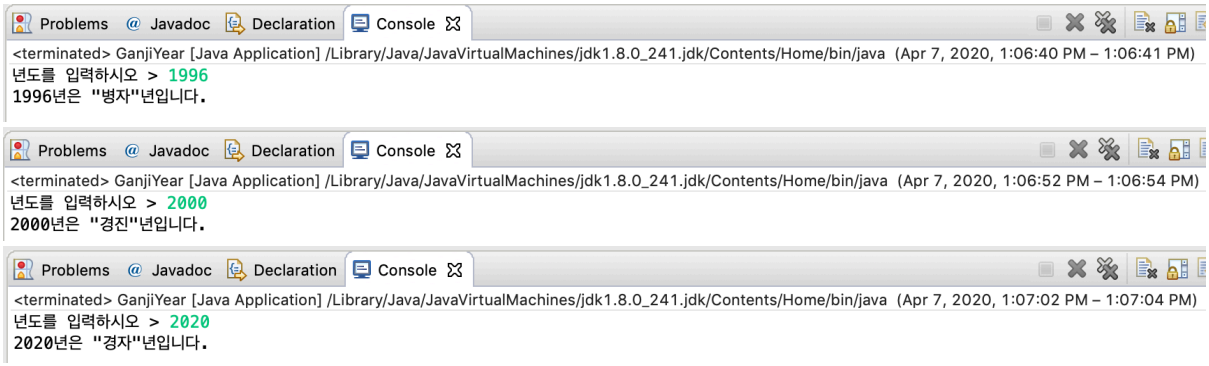
나머지	0	1	2	3	4	5	6	7	8	9
십간	경	신	임	계	갑	을	병	정	무	기

나머지	0	1	2	3	4	5	6	7	8	9	10	11
십이지	신	유	술	해	자	축	인	묘	진	사	오	미

D. 소스코드 및 주석

```
/*
 파일명: Ganji.java
 입력: 60갑자로 계산하고자 하는 년도
 출력: 입력 받은 년도의 간지
*/

import java.util.Scanner;
public class GanjiYear {

	public static void main(String[] args) {
		int year; // 사용자에게 입력 받을 년도
		char gan[] = {'경','신','임','계','갑','을','병','정','무','기'}; // gan 배열에 십간에 대한 10개 데이터 입력
		char ji[] = {'신','유','술','해','자','축','인','묘','진','사','오','미'}; // ji 배열에 십이지에 대한 12개 데이터 입력
		int i,j; // 간지를 계산할때 사용할 임시 변수

		Scanner scanner = new Scanner(System.in);
		System.out.print("년도를 입력하시오 > ");
		year = scanner.nextInt(); // 계산할 년도를 입력받음

		i = year % 10; // 입력받은 년도를 십간의 개수인 10으로 나눈 나머지 -> 열개 중 몇 번째인지 파악
		j = year % 12; // 입력받은 년도를 십이지의 개수인 12으로 나눈 나머지 -> 열두개 중 몇 번째인지 파악

		System.out.print(year + "년은 ₩"");
		System.out.print(gan[i]); // 위에서 구한 i로 십간 중 해당하는 문자 출력
		System.out.print(ji[j]); // 위에서 구한 j로 십이지 중 해당하는 문자 출력
		System.out.print("₩"년입니다.");

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

E. 입력 및 출력 결과 예시



3. 비정방형 2 차원 배열 출력하기

A. 문제

불규칙한 비정방형 2 차원 배열을 생성하여 정수를 입력 받아 저장하려고 한다.
아래 실행 예를 참고하여 동일한 결과를 나오도록 프로그램을 쓰시오.
열의 개수와 데이터 입력 시 첫번째 정수는 열의 크기를 의미하고 두번째 정수부터는 데이터이다.
비정방형 배열의 출력 결과를 나타낼 때는 while 문과 for-each 문을 이용하여 출력하시오.

```
배열의 정보를 입력하시오.
행의 개수> 3
열의 개수와 데이터를 입력하시오>
2 20 30
3 100 200 300
4 1 2 3 4
불규칙한 배열의 출력 결과
20 30
100 200 300
1 2 3 4
```

B. 해결 방법

1. 각각 행과 열을 나타내는 변수 ‘row’와 ‘column’을 선언한다.
2. 행의 수를 입력 받아 ‘row’에 저장한다.
3. ‘row’ 만큼의 행이 있는 2 차원 배열을 생성한다.
4. for 문을 사용해 위에서 입력 받은 행의 수 만큼 반복하여 다음과 같은 작업을 한다.
   a. i 번째 행의 열의 개수를 입력 받는다.
   b. for 문을 사용해 열의 개수만큼 데이터를 입력 받고 배열에 저장한다.
5. 모든 행과 열의 데이터를 입력 받은 후, while 문과 for-each 문을 사용해 결과를 출력한다.

C. 특이사항

- 출력 시 for-each 문을 통해 각 열의 데이터를 출력하고, while 문을 통해 각 행마다 줄을 바꾸어 준다.

D. 소스코드 및 주석

```
/*
 파일명: IrregularArray.java
 입력: 배열 행의 개수, 열의 개수 및 각 열에 저장할 데이터
 출력: 비정방형 배열
*/

import java.util.Scanner;

public class IrregularArray {

	public static void main(String[] args) {

		int row, column; // 행과 열 변수 선언

		System.out.println("배열의 정보를 입력하시오.");
		Scanner scanner = new Scanner(System.in); // 스캐너 생성
		System.out.print("행의 개수 > ");
		row = scanner.nextInt(); // 행의 개수 입력받기

		int array[][] = new int[row][]; // 입력받은 row를 통해 row개 만큼의 행이 있는 이차원 배열 만들어줌

		System.out.println("열의 개수와 데이터를 입력하시오 > ");

		for(int i=0;i<row;i++) { // 이하를 첫번째 행부터 마지막 행까지, 행의 개수 만큼 반복할 것
			column = scanner.nextInt();  // 해당 행의 열의 개수 입력받기
			array[i]=new int[column]; // 해당 행에 입력받은 column을 통해 column개 만큼의 열 공간을 만들어줌
			for(int j=0;j<column;j++) { // 이하를 위에서 입력받은 column의 개수 만큼 반복할 것
				array[i][j] = scanner.nextInt(); // 해당 열의 각 행에 저장할 데이터를 순서대로 입력받음
			}
		}

		int i=0; // 임시변수

		while(i<array.length) { // 위 배열 행의 개수만큼 반복
			for(int x : array[i]) { // for-each 문을 통해 각 행의 데이터 출력하기
				System.out.print(x);
				System.out.print(" ");
			}
			System.out.println(); // 한 행의 출력을 마칠 때마다 줄 넘김
			i++; // 다음 행으로 넘어감
		}

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

E. 입력 및 출력 결과 예시

4. 소수 50 개 출력하기

A. 문제

Write a program that displays the first 50 prime numbers in five lines, each of which contains 10 numbers.
소수의 첫 50 개를 각 10 개의 숫자로 이루어진 다섯 줄로 나타내는 프로그램을 작성하시오.

B. 해결 방법

1. 총 50 개의 소수를 저장할 1 차원 배열 'prime'을 생성한다.
2. 조건문에서 활용하기 위한 임시 변수 'temp'를 선언하고 1 로 초기화한다.
3. 검사할 숫자를 나타내는 변수 'num'을 선언하고, 2 를 저장한다.
4. for 문을 사용하여 임시 변수 i 가 50 이 될 때까지 이하를 반복한다.
   여기서 i 는 소수를 하나씩 구할 때 마다 증가하게 하고, 소수를 구하면 배열의 i 번째 자리에 저장한다.
   a. 이중 for 문을 통해 검사할 숫자를 2 부터 그의 제곱근까지의 모든 정수로 나누어 나온 나머지를 계산한다.
   b. 나머지 값이 0 일 경우, temp 의 값을 0 으로 바꾸고 break 를 통해 이중 for 문에서 빠져나온다.
   c. if 문을 통해 temp 가 0 으로 바뀐 경우를 걸러준다. 바뀌지 않고 1 일 경우 num 을 배열의 i 번째 자리에 저장한 후 i 를 1 씩 증가한다.
   d. temp 의 값을 다시 1 로 초기화한다.
5. 4 번의 반복문을 거쳐 i 가 50 이 되면, 배열 prime 에 50 개의 값이 저장된 것 이므로 출력을 실행한다.
6. 출력 시 for 문을 통해 배열의 0~49 번째 자리에 저장된 데이터를 각각 출력하되, 이중 for 문을 통해 데이터 10 개 출력 마다 줄 바꿈을 해준다.

C. 특이사항

- 소수가 아닌 수는 제곱근 이하의 수로 나누어 떨어진다. 따라서, 각 숫자를 자신 이하의 모든 수로 나눌 필요 없이, 자신의 제곱근 까지만 나누어 주는 것이 효율적이다.
- 소수는 1 과 자신을 제외한 수로 나누어 떨어질 수 없다. 즉, 모든 수는 1 로 나누어 떨어지므로 나누는 수는 2 부터 시작한다.
- 1 은 소수가 아니므로 검사할 숫자인 'num'역시 2 부터 시작한다.

D. 소스코드 및 주석

```
/*
 파일명: PrimeNumber.java
 입력: -
 출력: 가장 작은 소수부터 50개를 10개씩 50줄로 총 50개 출력
 */

public class PrimeNumber {

	public static void main(String[] args) {

		int[] prime = new int[50]; // 50개의 데이터를 저장할 수 있는 배열 prime 생성
		int temp = 1; // 조건문을 위해 임시로 설정한 변수
		int num = 2; // 소수인지 검사를 진행하게 될 숫자에 대한 변수 선언, 1을 빼고 2부터 시작

		for(int i=0; i<50; num++) { // 50개의 소수를 찾아낼때까지 반복하기 위해 임시로 변수 i 선언, 검사할 숫자 num을 1씩 증가하며 반복
			for(int j= 2; j<=Math.sqrt(num); j++) { // 검사하려는 숫자의 제곱근 만큼 반복, j가 2부터 제곱근일때까지 1씩 커짐
				if( num%j == 0 ) { // 검사숫자가 j로 나누어 떨어질 경우 -> 1과 자기자신을 제외한 숫자로 나눌 수 있으므로
소수 아님
					temp = 0; // 밑 조건문에 걸리지 않도록 temp를 0으로 변경
					break; // break로 인해 반복문에서 빠져나옴
					}
				}
			if (temp != 0) { // temp가 0으로 바뀌지 않았다면 위 if문에서 나누어 떨어지는 숫자가 없었다는 것이므로 소수
				prime[i]=num; // i번째 소수! prime 배열의 i번째 자리에 검사한 숫자 num 저장
				i++; // 다음 소수를 찾았을 때 배열의 올바른 자리에 저장하기 위해 i에 1 증가
				}
			temp = 1; // 다시 temp를 1로 초기화
		}

		// 모든 50개의 소수를 배열 prime에 저장하였으니, 데이터 10개씩 다섯 줄 출력하기

		for(int i=0; i<50; ) { // 배열 [0]자리에 있는 데이터 부터 [49]까지모두 출력할 때 까지 반복
			for(int j=0; j<10; j++) { // 열 개의 데이터 출력 후 줄 바꾸기 위한 반복문
				System.out.print(prime[i]+" "); // 순서대로 데이터 출력 후 띄어쓰기
				i++; // 다음 데이터 출력을 위해 i 에 1 증가
			}
			System.out.println(); // 열 개 출력 후 줄 바꿈
		}

	}
}
```

E. 입력 및 출력 결과 예시