8장 연습문제

1. 다음과 같이 함수p를 호출할 때 값 전달, 참조 전달, 값-결과 전달, 이름 전달 방법을 사용했을 각 경우에 대해 호출 후의a의 값은 무엇인가?

```
int a = 1;
void p(int x, int y)
{
    x++
;
    y++
;
}

main()
{
    p(a, a);
}
```

(1)값 전달
1
(2)참조 전달
3
(3)값-결과 전달
2
(4)이름 전달
하지 않아도 됨

2. 매개변수 전달 방법으로 값 전달, 참조 전달, 값-결과 전달, 이름 전달 방법을 사용할 때 이 프로그램의 출력은 각각 무엇인가?

```
int i = 1;
int a[3] = {1, 2, 3};

void swap(int x, int y)
{
    int t = x;
    x = y;
    y = t;
}

main()
```

```
{
    swap(i, a[i]);
    printf("%d %d %d %d\n", i, a[0], a[1], a[2]);
}
```

(1)값 전달
1123
(2)참조 전달
2113
(3)값-결과 전달
2113
(4)이름 전달
하지 않아도 됨

3. 정적 유효범위 법칙과 동적 유효범위 법칙을 사용했을 때 다음 프로그램의 실행결과를 설명하시오.

```
int x= 0;
fun void g(int y)
      x=x+ y * y;
fun void f(int z)
      let int x= 10; in
            g(z);
            print x;
      end;
f(5);
print x;
```

(1)정적 유효범위
25 25

(2)동적 유효범위
35 0

4. 함수, 프로시저, 메소드, 멤버 함수 등의 용어에 대해서 비교 설명하시오.

함수, 프로시저, 메소드, 멤버 함수는 모두 함수를 나타내는 용어이다.
여기서 함수라는 용어는 반환 값이 있을 때 주로 사용된다.
프로시저는 반환 값이 없을 때 많이 사용되는 용어이다.
메소드는Java와 같은 객체지향 언어에서 일반적으로 많이 사용되는 용어이며 멤버 함수는C++에서 사용되는 용어이다.

5. 형식 매개변수, 실 매개변수, 인자, 지역변수 등의 용어에 대해 비교 설명하시오.

형식 매개변수는 함수 정의에서 선언된 매개변수를 나타낸다.
실 매개변수는 함수 호출에서 사용된 매개 변수로 인자라고도 한다.
지역 변수는 함수 내에서 선언된 변수를 나타낸다.

6. 4가지 매개변수 전달 방법과 이 방법을 지원하는 언어에 대해서 설명하시오.
(생략)

7. 다음 용어에 대해서 예를 들어 설명하시오.
(1)바인딩
(2)정적 속성과 동적 속성
(3)정적 바인딩과 동적 바인딩
272, 273쪽 참조

8. C언어와Java언어에서 다음 속성에 대한 바인딩 시간은 언제인가?
(1) char의 의미
C : 언어 정의 시간
Java : 언어 정의 시간

(2) 배열의 크기
C : 컴파일 시간
Java : 실행 시간

(3) 지역변수의 메모리 위치
C : 실행 시간
Java : 실행 시간

(4) 이름 상수의 값
C : 컴파일 시간
Java : 컴파일 시간

(5) 함수의 시작 위치
C : 컴파일 시간
Java : 컴파일 시간

9장 연습문제

1. 함수 구현을 위한 실행시간 스택과 활성 레코드에 대해 설명하시오.
296, 297쪽 참조

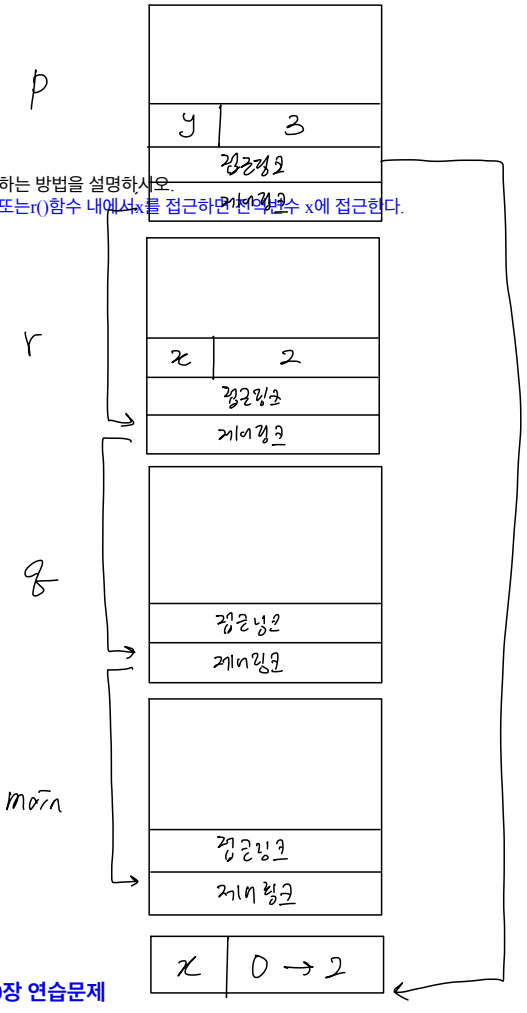
2. 제어 링크와 접근 링크에 대해서 설명하시오.
접근 링크를 이용한 비지역 변수 접근에 대해서 설명하시오.
311쪽 참조

4. 다음C프로그램에 대해 답하시오.

(2) 이 프로그램의 출력 결과는 무엇인가?
3, 2
3, 2

(3) 함수p()가 호출되었을 때 변수x와y를 접근하는 방법을 설명하시오.
C언어는 정적 유효범위 규칙을 사용하므로p()또는r()함수 내에서x를 접근하면 전역변수 x에 접근한다.
y는 지역 변수를 접근한다.

```
int x = 0;
void p( ) {
      int y = 3;
      printf("%d\n", y);
      printf("%d\n", x);
}

void r() {
      x = 2;
      p();
}

void q() {
      int x = 1;

      r();          (1)  3, 2

      p();          (2)  3, 2
}

int main() {
     q();
     return 0;
}
```



10장 연습문제

1. 예외 처리 방법으로 재개 모델과 종료 모델에 대해서 설명하시오.
321쪽 참조

2. Java, C++및Python언어의 예외 전파 과정에 대해 설명하시오.
330,339쪽 참조

3. Java언어에서 예외 검사를 하는 이유는 무엇인가? 예외 검사 과정에 대해 설명하시오.

Java언어에서 예외 검사를 하는 이유는 예외 처리를 강제하여 프로그램의 안정성을 높이기 위해서이다. 예외 검사를 통해 예외 처리 상황을 미리 예측하고, 그 상황에 대한 처리 방법을 정의함으로써 예기치 않은 오류를 방지할 수 있다.

Java에서 예외 처리는 두 가지 방법으로 이루어진다.

첫 번째 방법은try-catch블록을 사용하여 예외를 처리하는 방법이다. try-catch블록은try블록 안에서 예외가 발생할 가능성이 있는 코드를 작성하고, catch블록에서 해당 예외를 처리하는 코드를 작성한다. 만약 예외가 발생하면, try블록 안의 코드는 즉시 중단되고, 해당 예외를 처리하는 catch블록으로 제어 흐름이 이동한다.

두 번째 방법은throws키워드를 사용하여 예외를 호출자에게 전달하는 방법이다. 메서드에서 발생하는 예외를 throws키워드를 사용하여 호출자에게 전달함으로써, 호출자에서 해당 예외를 처리할 수 있도록 한다. 예외를 throws한다는 것은 해당 메서드를 호출하는 쪽에서 예외 처리를 하도록 강제한다는 의미입니다.

예외 검사는 이러한 예외 처리가 제대로 수행되는지 검사한다. 그 과정은 다음과 같다.
(1) 예외가 발생할 가능성이 있는 코드가try블록 안에 작성되었는지 검사합니다.
(2) 예외가 발생한 경우, 해당 예외를 처리하는catch블록이 작성되었는지 검사한다.
(3) 해당 예외를 처리하는catch블록이 없으면 해당 예외가 메소드 헤더에 throws에 의해 선언되어 있는지 검사한다.
(4) 메소드 내에서 다른 메소드를 호출하는 경우에는 피호출자 메소드 헤더에throws절로 선언된 예외 정보를 참조하여 이 예외를 처리할 수 있는try-catch문이 있는지 검사하고 없으면 호출자 메소드 헤더에 이 예외가 선언되어 있는지 검사한다.

4. Java언어의 검사 예외와 비검사 예외에 대해서 설명하시오. 비검사 예외를 컴파일 시간에 예외 검사하지 않는 이유는 무엇인가?

검사 예외(Checked Exception)는 컴파일러에 의해 체크되며, 예외 처리를 강제한다. 이러한 예외는try-catch블록을 사용하여 예외 처리를 하거나throws키워드를 사용하여 예외를 호출자에게 전달해야 한다. 주요 예외로는 IOException, SQLException등이 있다.

반면, 비검사 예외(Unchecked Exception)는 컴파일러에 의해 체크되지 않는다. 이러한 예외는 RuntimeException클래스와 그 하위 클래스들에 속한다. 예외 처리를 강제하지 않으며, 개발자가 필요에 따라 예외 처리를 선택적으로 할 수 있다. 주요 예외로는 NullPointerException, ArithmeticException등이 있다.

검사 예외는 프로그램에서 일어날 수 있는 예외 상황을 미리 예측하여 처리할 수 있도록 하여 프로그램의 안정성을 높이는데 사용된다. 반면, 비검사 예외는 보통 프로그램의 버그나 잘못된 사용에 의해 발생하므로, 이러한 예외가 발생하는 경우에는 개발자가 이를 책임지고 수정해야 한다.

5. 다음 프로그램의 실행 결과는 무엇인가?
(0으로 나눗셈을 할 때 ArithmeticException이 발생함)

```
public class Propagate1 {
    static void divide (int adjustment) {
        int current = 1;
        System.out.println("1");
        current = current / adjustment;
        System.out.println("2");
    }
    static void via() {
        System.out.println("3");
        divide (0);
        System.out.println("4");
    }
    public static void main() {
        System.out.println("5");
        try {
            via();
        } catch (ArithmeticException m) {
            System.out.println("6");
        }
        System.out.println("7");
    }
}
```

5
3
1
6
7