18-2 자바프로그래밍 중간

1. 자바언어의 특징 3가지 쓰기 (주관식)

1) 플랫폼 독립성: jvm만 설치하면 어디서든 바이트코드를 실행할 수 있음
2) 객체 지향: 객체 단위로 구현해 프로그램 구성, 높은 코드 재활용성과 정확성, 신속성
3) 언어 차원의 멀티스레딩: 동일한 프로그램 내에서 여러 개의 작업 동시 진행 가능

2. 오류가 나는 것 찾기 (객관식 : 4지선다)

1) byte x = 290 → 1byte (-128 ~ 127)

2) int y = 40000 → 4byte

3) short z = -30000 → 2byte (-32768 ~ 32767)

4) double w = 77 → 8byte

자료형	설명	크기(바이트)	범위
byte	부호있는 정수	1바이트	-128에서 127
short	부호있는 정수	2바이트	-32768에서 32767
int	부호있는 정수	4바이트	-2147483648에서 2147483647(20억 정도)
long	부호있는 정수	8바이트	-9223372036854775808에서 9223372036854775807
float	부동소수점형	4바이트	약 ± 3.40282347 × 10^{+38}(유효숫자 6-7개 정도)
double	부동소수점형	8바이트	약 ± 1.7976931 × 10^{+308}(유효숫자 15개 정도)
char	문자형	2바이트	\u0000에서 \uFFFF
boolean	논리형	1비트	true, false

3. 계산 값 쓰기 (주관식)

1) 1 | 2 = ? 3

2) 1 + 2 & 5 = 1 (연산자 순서는 맞지만 숫자는 부정확)

3) 20 << 2 = 80

4) -5 >>> 2 = ?

몇바이트기준?

1) 0001 | 0010 → 0011 → 3

2) 0010 & 0101 → 0000 → 0

3) 10100 → 1010000 → 80

4) 5는 0101, -5는 1011 → 0001

4. 식별자로 쓸 수 없는 것 모두 고르기

**필요했던 개념 : 숫자는 앞에 못옴, 특수기호 $ 사용안됨

알파벳 or _ 로시작, 숫자가능, 특수문자는 _ 와 $만가능, 한글가능(유니코드)

5. 결과값 쓰기

```
int  m = 2;
int n = 1'
switch(m) {
   case 1 : n--; break;
   case 2 : m++;
   case 3 : m++;
}
system.out.println(m); → 4
```

6. 이름이 two이고 크기는 5., 7인 이중배열 선언하고, 행과 열 인덱스 값의 합을 저장하는 코드 쓰기

```
int [][] two = new int[5][7];
int sum = 0;
for (int i = 0; i < two.length; i++) {
   for (int j = 0; j < two[i].length; j++)
      sum += two[i][j];
}
```

7. Args 인자로 들어온 정수 두개의 합 구하는 방법

```
Class Test {

  Public static void main(String[] args){

      int sum = Integer.parseInt(args[0]) + Integer.parseInt(args[1]);
      System.out.println(sum);
```

```
    }
}
```

8. 캡슐화 개념과 하는 방법 쓰기

관련된 데이터와 알고리즘을 하나의 묶음으로 묶는 것
목적: 서로 관련된 데이터와 알고리즘을 묶어 관리를 용이하게 함, 객체를 캡슐로 싸서 내부를 보호함
정보 은닉을 이용함 -> 공개된 인터페이스를 통해서만 객체에 접근 가능하도록 함

9. 생성자 특징 3가지

1) 개발자가 아무것도 정의하지 않으면 매개변수가 없는 생성자가 자동 생성됨
2) 오버로딩: 매개변수가 다른 동일한 이름의 생성자를 여러 개 만들 수 있음
3) this()를 이용해 생성자 내부에서 다른 생성자 호출 가능
4) 반환값이 없는, 클래스 이름과 동일한 메서드

10. 다중상속이 무엇인가? 자바는 다중상속을 지원하는가?

하나의 클래스가 여러 개의 부모를 갖는 것
자바는 다중상속을 지원하지 않는다.

11. 오류가 있다면 고치고, 없다면 출력결과 쓰기

```
Class C1{
   C1() {
      System.out.println(“class c1“);
   }
}
Class C2 extends c1{
   C2(int x) {
      System.out.println(“class c2“);
   }
}
Class test {
   Public static void main(String [] args){
   C2 construc = new C2(5);
   }
}
```

오류 없음
출력결과
class c1
class c2

12. overloading, overriding 비교 서술

오버로딩: 매개변수가 다르며 동일한 이름의 메서드를 여러 개 작성하는 것
오버라이딩: 자식 클래스가 부모 클래스의 메서드를 재정의하는 것
공통점: 이름을 재사용함
차이점: 오버로딩은 컴파일시간에서의 다형성 지원, 오버라이딩은 실행시간에서의 다형성 지원

13. 상속의 장점

부모의 속성과 동작을 물려받기 때문에 코드의 재사용성이 높아지고 더 빠르고 정확한 코딩이 가능함
자바의 특징 중 하나인 다형성을 가능하게 함
-> 부모의 메서드를 재정의해 사용하는 오버라이딩

14. 6장 예제 6-7 String 클래스 메소드 활용문제

15. Interface & class : 조건 주고 코드 전체 짜기 (Interface : Shape, Class : Rectangle, Circle)

16. super class & sub class : 조건 주고 코드 전체 짜기 (super class : Person, sub class : Student)