1. 컴파일러와 자바 가상 기계에 대해 설명하시오.

자바는 다양한 컴퓨터에서 동일한 모습으로 실행 가능 한데 그 이유는 바로 자바 가상

기계 때문이다. 자바 컴파일러는 특정한 컴퓨터가 아닌 가상적인 기계를 위한 코드를 생성한다.

예시)

Hello.java -> **①컴파일러** -> Hello.class -> **②자바가상기계** -> Hello

① 컴파일러

- 자바 프로그램을 가상 기계의 명령어로 반환

- 소스 코드를 컴파일 하면 바이트 코드가 생성

② 자바가상기계

- 가상 기계의 명령어를 실제 기계의 명령어(0101001)로 변환

- 바이트 코드를 해석하여 실행하는 소프트웨어

** 자바는 특정한 컴퓨터가 아닌 중간적인 코드를 생성하고 이것을 해석하여 실행하는 구조로 되어 있기 때문에 어떤 컴퓨터에서나 실행 가능 하다.*

** 자바는 바이트 코드와 가상 기계 때문에 컴퓨터 구조에 중립적이다.*

2. JDK와 JRE에 대해 설명하시오.

JRE(Java Runtime Environment)

- JRE는 자바 프로그램을 실행하기 위한 라이브러리, 자바 가상 기계, 기타 컴포넌트 제공

- 자바 프로그램을 실행만 시킬 수 있는 환경

JDK(Java Development Kit)

- JDK는 JRE에 추가로 자바 프로그램을 개발하는 필요한 컴파일러, 디버거와 같은 명령어행 개발 도구를 추가
- 개발도 할 수 있는 환경

- 자바프로그램을 개발 하는 개발 도구

** JDK = JRE + 컴파일러 + 디버거 등*

3. 자바 프로그램 개발 단계를 설명하시오.

* 소스파일 생성 -> 컴파일 -> 클래스 적재 -> 바이트 코드 검증 -> 실행

① 소스 파일 생성

- 윈도우 보조 프로그램의 메모장 사용

- 파일 확장자 : .java

② 컴파일

- .java(소스파일) -> .class(클래스) 소스파일을 컴파일 하여 클래스 파일로 변환

- javac

③ 클래스 적재

④ 바이트 코드 검증

⑤ 실행

- 자바 기계 상에서 클래스 파일 실행(java)

* java 명령어가 바로 "자바 가상 기계" 구현

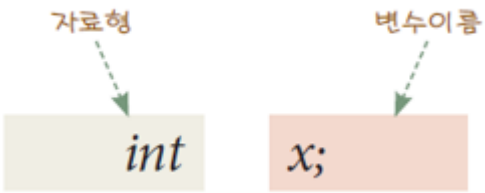
3. 변수와 상수에 대해 설명하시오.

1) 변수(variable)

- 프로그램이 사용하는 데이터를 일시적 저장할 목적으로 사용하는 메모리 공간

- 데이터를 저장할 수 있는 상자

- 변수에도 여러 가지 타입 有



2) 상수(constant)

- 그 값이 프로그램이 실행하는 동안 변하지 않는 수

- 예) 3.14, "Hello World"

** 변수는 실행 도중 값 변경 가능, 상수는 한번 값이 정해지면 변경 불가능*

4. 자료형에 대해 설명하시오.

* 자료형(data type) : 자료의 타입

- 물건을 정리하는 상자도 다양한 타입이 있듯이 자료를 저장하는 변수도 다양한 종류가 有

1) 기초형

- 실제 값이 저장

① 정수형 : byte, short, int, long

② 실수형 : float, double

③ 논리형 : boolean

④ 문자형 : char

2) 참조형

- 실제 객체를 가리키는 주소 저장

클래스, 인터페이스, 배열

5. switch 문에 대해 설명하시오.

- 여러가지 경우 중에서 하나를 선택하는 데 사용

- if 문과 비슷하지만 좀 더 정형화된 조건 판단문

* 구조

```
switch (입력 변수) {
    case 입력 값1;
        break;
    case 입력 값2;
        break;
    ...
    default : ...
        break;
}
```

- case 문 마다 break 라는 문장이 있는데 case문을 실행 한 뒤 switch 문을 빠져나가기 위한 것

- 만약 break 문이 빠져 있다면 그 다음의 case 문이 실행

6. 반목문 종류와 반복 구조의 필요성에 대해 설명하시오.

1) 반복문 종류

while

```
while (조건식) {
    반복문장
}
```

- 주어진 조건이 만족되는 동안 문장들을 반복 실행

1-1. 무한반복

```
while(true){
    ...
    // 반복되는 코드를 적는다.
}
```

do-while 문

```
do {
    반복문장;
} while (조건식);
```

- 반복 조건을 루프의 끝에서 검사

1. 문장들이 실행

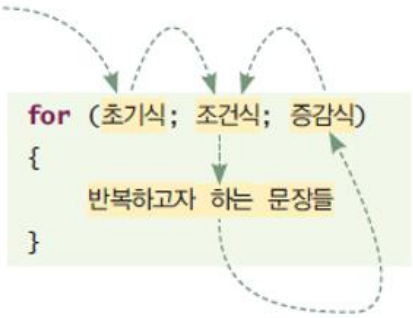
2. 조건식 계산

3. 결과가 참이면 1로 돌아감

4. 결과가 거짓이면 종료

for 문

- 정해진 횟수 만큼 반복하는 구조



1. 초기화 실행

2. 반복 조건을 나타내는 조건식 계산

3. 수식의 값이 거짓이면 for 문 실행 종료

4. 수식의 값이 참이면 문장 실행

5. 증감 실행하고 2로 돌아감

2) 반복 구조가 왜 필요한가?

- 같은 처리 과정을 되풀이 하는 것이 필요하기 때문이다. 예를 들어 학생 40명의 평균 성적을 구하려면 같은 과정을 40번 반복해야 한다.

7. break와 continue의 차이점을 설명하시오.

1) break문

- 자신이 포함된 가장 가까운 반복문을 벗어난다

2) continue문

- 반복문의 끝으로 이동하여 다음 반복으로 넘어간다. break문과 다르게 ***반복문 전체를 벗어나지 않고*** 다음 반복을 계속 수행할 수 있다. 즉 for( ) { } 구문을 벗어나지 않고 구문의 끝으로 이동하여 조건식이 false가 될 때 까지 반복한다.

* 각각 언제 사용하는 게 좋은가?

- break문은 특정 조건을 만족하면 반복뭇을 벗어날 때 사용하면 유용,

continue문은 전체 반복 중 특정 조건을 만족하는 경우를 제외하고자 할 때 유용하다.

8. 절차 지향과 객체 지향의 차이점을 설명하시오.

- 실제 세계를 모델링하여 소프트 웨어를 개발하는 방법

1-1. 절차 지향 procedural programming

- 문제를 해결하는 절차를 중요하게 생각하는 방법

- 데이터와 알고리즘이 묶여 있지 않다.

1-2. 객체 지향 Object-Oriented Programming

- 데이터와 절차를 하나의 덩어리(객체)로 묶어서 생각하는 방법

- 컴퓨터 하드웨어 부품을 구입하여서 컴퓨터를 조립하는 것과 유사하다.

- 데이터와 알고리즘이 묶여 있다.

9. 객체 지향의 3대 구성요소와 장점을 설명하시오.

1-1. 캡슐화 encapsulation

- 관련된 데이터와 알고리즘(코드)이 하나의 묶음으로 정리

- 오류가 없어서 사용하기 편리

- 데이터를 감추고 외부 세계와의 상호작용은 메소드를 통하는 방법

- 업그레이드가 쉬움(라이브러리가 업그레이드 되면 쉽게 변경 가능, 정보 은닉 가능)

1-2. 상속 inheritance

- 이미 작성된 클래스(부모 클래스)를 이어 받아 새로운 클래스(자식 클래스)를 생성하는 기법

- 기존의 코드를 재활용, 중복 최소화(중복되는 코드는 수퍼 클래스에 모은다)

- is-a 관계

1-3. 다형성 polymorphsim

- 하나의 이름(방법)으로 많은 상황에 대처하는 기법

- 동일한 작업을 하는 멤버 함수들에 똑같은 이름을 부여 할 수 있으므로 코드가 더 간단

- 객체들의 타입이 다르면 똑같은 메세지가 전달되더라도 서로 다른 동작을 하는 것

- 부모 클래스에 해당되는 animal 클래스에 dog, cat 등 모든 동물이 해당되는 것들을 대입 가능

=> 무엇이 대입이 되든 간에 부모는 해당 동물의 짖는 소리를 듣고 싶으면 speak()만 호출하면 된다!?

- 메소드의 매개 변수로 수퍼 클래스 참조 변수를 이용한다 => 다형성을 이용하는 전형적인 방법

2. 장점

- 신뢰성 있는 소프트 웨어를 쉽게 작성

- 코드 재사용 쉬움

- 업그레이드 쉬움

- 디버깅 쉬움

10. 클래스에 대해 설명하시오.

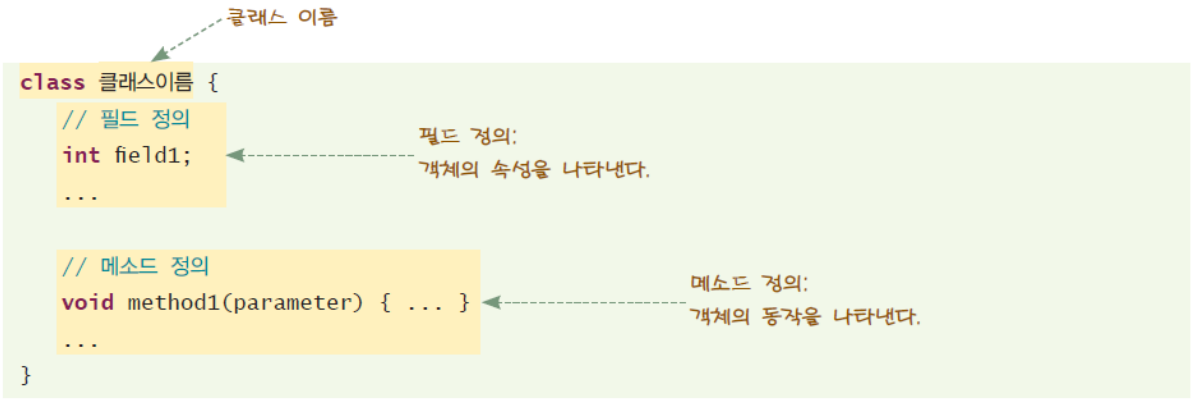
- 클래스 Class : 객체를 만드는 설계도

- 클래스로 부터 만들어지는 각각의 객체를 특별히 그 클래스의 인스턴스(instance) 라고도 함

- 예를 들어 사람이 도면을 그리고 그 도면들로 자동차를 만드는데 여기서 도면은 class, 자동차는 object를 가리킨다.

- 클래스는 객체를 찍어내는 틀과 같다!

* 클래스의 구조



11. 객체를 생성하는 방법에 대해 설명하시오.

- 객체는 **속성**과 **동작**을 가지고 있다.

```
Car myCar = new Car( );
```

- 자바에서 객체를 생성하는 방법은 new 한가지 뿐이다.

```
Car    myCar;          // ❶ 참조 변수를 선언
myCar  =  new  Car();  // ❷ 객체를 생성하고 ❸ 참조값을 myCar에 저장
```

① 참조 변수 선언

- Car 타입의 객체를 참조할 수 있는 변수 myCar 선언

② 객체 생성

- new 연산자를 이용하여 객체 생성 및 객체 참조값 반환

③ 참조 변수와 객체 연결

- 생성된 새로운 객체의 참조값을 myCar 라는 참조 변수에 대입

** 주의

```
Car myCar;
```

- 위의 문장은 객체가 생성된 것이 아니다

- 객체를 가리키는 참조값을 담을 수 있는 변수만 생성된 것!

12. 기초 변수와 참조 변수 차이점

- 기초 변수는 할당 메모리에 저장되는 데이터가 저장되는 반면, 참조 변수는 해당 주소 값이 저장된다. 즉, 처음 객체 선언해서 초기화 하면 해당 객체는 주소 값이 저장되어 있다.

13. 변수의 종류와 개념을 설명하시오.

1) 필드 Field

- 클래스 안에서 선언되는 멤버 변수, 인스턴스 변수라고 함

* 필드의 선언

접근 지정자	필드의 타입	필드의 이름(첫 글자만 소문자)
public	int	speed;

1-1) 설정자와 접근자

- 설정자 mutator : 필드의 값 설정하는 메소드(setXXX())

- 접근자 accessor : 필드의 값 반환하는 메소드(getXXX())

* 접근자와 설정자 메소드만을 통하여 필드에 접근

* 설정자와 접근자를 사용하는 이유

- 설정자에서 매개 변수를 통하여 잘못된 값이 넘어오는 경우, 이를 사전에 차단 가능

- 필요할 때 마다 필드값을 계산하여 반환 가능

- 접근자만을 제공하여 자동적으로 읽기만 가능한 필드를 만들 수 있음

2) 지역 변수 local variable

- 메소드나 블록 안에서 선언되는 변수

- 메소드의 매개변수도 지역 변수의 일종

* 주의 : 지역 변수를 초기화 하지 않고 사용하면 오류 발생

3) 매개 변수 parameter

- 메소드 안에서의 변수로 사용

- 매개 변수가 기초형의 변수 일 경우, 값이 복사된다.

- 매개 변수가 참조형의 변수 일 경우도 참조값이 복사된다 **(중요!!!)**

14. 정적 메소드에 대해 설명하시오.

- 정적 메소드 static method : 객체를 생성하지 않고 사용할 수 있는 메소드

예시) Math 클래스에 들어 있는 각종 수학 메소드

```
double value = Math.sqrt(9.0);
```

- 정적 메소드에서는 인스턴스 변수와 인스턴스 메소드에 접근 할 수 없다

Why? 객체가 생성되지 않았으므로 인스턴스 메소드는 존재 無

- 상수는 정적 변수로 만들어서 공유하는 것이 메모리 공간을 절약한다.

- 정적 변수를 외부에서 사용할 때 "클래스이름.정적변수" 형식 사용

15. 접근제어에 대해 설명하시오.

- 접근제어 access control

: 다른 클래스가 특정한 필드나 메소드에 접근 하는 것을 제어

1) Public

- 다른 모든 클래스가 사용할 수 있는 클래스

2) Package

- 수식자가 없으면 같은 패키지 안에 있는 클래스만이 사용

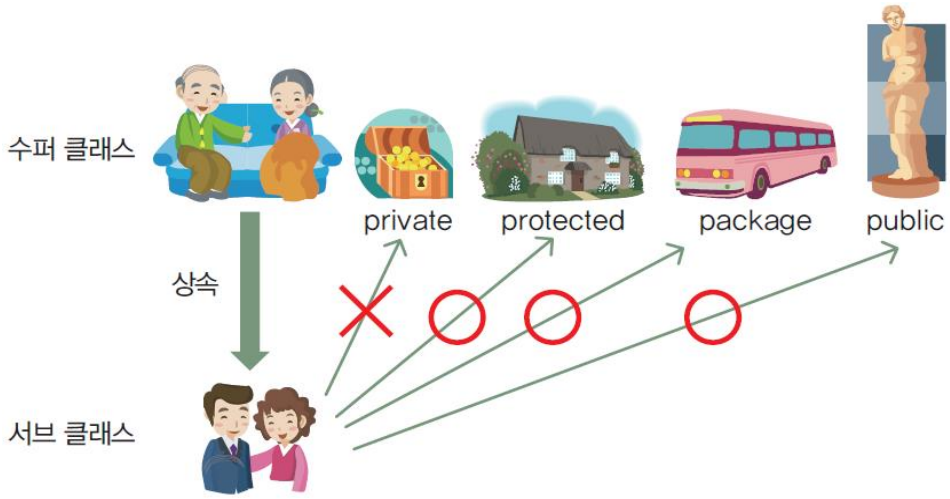
- 같은 디렉토리? 안에 있는 것

3) Protected

- 다른 패키지에 있더라도 상속 받은 자식 클래스는 부모 클래스에 있는 것들 사용

4) Private

- 해당 클래스 내에서만 사용 가능



* 접근 범위

private < package < protected < public

16. 추상 클래스와 인터페이스에 대해 각각 설명하시오.

1) 추상 클래스

- 추상 메소드를 가지고 있는 클래스

- new 연산자를 사용해서 인스턴스화 X -> 대신 abstract 키워드 이용

- 상속 시키기 위해 만든 클래스

- 추상 메서드가 없어도 추상 메소드라고 정의 가능(인터페이스는 안됨)

- 메서드에 이름, 리턴타입, 매개변수 만 정의하고 동작하는 실제 코드 無

- 추상 메서드가 하나라도 있으면 해당 클래스는 추상 클래스

- 추상 메소드가 아니였다면 자식 클래스에서 몸체를 만들지 않아도 되지만 추상 메소드는 반드시 자식 클래스에 구체적인 코드가 짜여 있어야 한다!(명시, 지시)

- 추상 메소드는 일반 메소드도 가질 수 있다.

* 추상 메소드를 만든 이유?

- 나를 상속 받은 어떤 클래스 든 간에 실질적인 move 라는 구체적인 행위는 각각 다르다.

move에 맞는 정의는 자기가 알아서 형식만 정해놓으면 되기 때문이다.(몸체부분은 자식 클래스가 알아서 구현해야 함)

- 상속 시킬 목적으로 만들어 졌다.

2) 인터페이스

- 클래스는 자바 내에 다중 상속이 안되는데 이 때 인터페이스를 사용하면 다중 상속이 가능

- 클래스와 클래스, 객체와 객체 간 사이에 접촉 할 수 있도록 해주는 통로 같은 것?

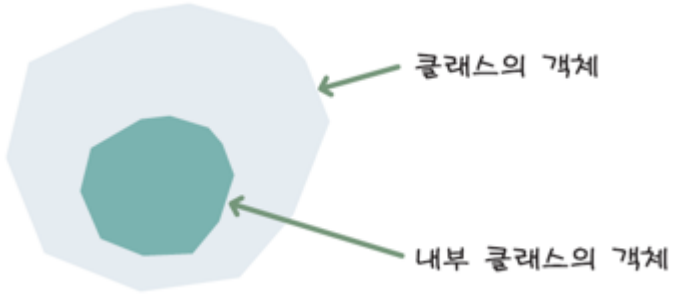
- 서로 연결하기 위한 객체와 객체 간의 상호작용

- implements 라는 키워드 사용(상속)

- 추상 클래스와 형태는 유사하나 추상 메서드만 가지고 있다. 즉, 몸체가 포함되어 있는 클래스가 없다

예시) JDBC(JavaDataBaseConectivite) -> 자바의 DB 같은 것

17. 내부 클래스에 대해 설명하시오.



- 내부 클래스 inner class : 클래스 안에 다른 클래스 정의

- 내부 클래스는 다른 클래스 내부에 정의된 클래스이다.

- 외부 클래스의 모든 멤버를 자유롭게 사용 가능

** 내부 클래스를 사용하는 이유?*

- 멤버 변수를 private로 유지하면서 자유롭게 사용 가능

- 하나의 장소에서만 사용되는 클래스들을 한 곳에 모으기 가능

- 보다 읽기 쉽고 유지 보스가 쉬운 코드