03

03-4. static과 메모리

Java 객체지향

“static” 키워드

- 클래스내에 선언된 멤버(변수, 함수)는 두가지로 구분
  - 객체 멤버
  - 클래스멤버

- 객체 멤버는 객체 생성할 때 메모리 할당
- 객체로 멤버 이용

- 클래스 멤버는 static 예약어로 선언

“static” 키워드

- static은 main 메서드가 실행되기 전에 메모리에 로드된다. 즉 main 실행 전에 정적 할당된다.
- static 메모리는 class별로 존재한다.
- static 멤버는 자바 프로그램이 실행되자 마자 메모리에 할당되며 종료될 때까지 메모리에 존재한다.
- static 메모리를 관리하는 방법은 없다.
- static 데이터는 최대한 적게 설계하는 것이 좋다.

- 프로그램이 실행되고, 프로그램이 종료될 때까지 유지되어야 하는 데이터
- 객체별로 공유되어야 하는 데이터
- static은 main 메서드가 실행되기 전에 메모리에 단 한번 로드되기 때문에 프로그램이 실행되는 동안 두 번 메모리에 로드하는 방법은 없다.

“static” 키워드

- “static” 변수
  - 해당 클래스의 전체 객체(Object) 간 공유되는 변수
  - 객체에 따라 값이 다르지 않음

```
class Car {
   String name;
   int speed;
   static int numberOfCars;

   public Car() {
          name = “MyCar” ;
          speed = 0;
          numberOfCars++;
   }

   public void setName(String name) {
          name = name;
   }
   public void setSpeed(int s) {
          speed = s;
   }
   public int getSpeed() {
          return speed;
   }
```

```
public class CarTest {
   public static void main()
   {
          Car mcqueen = new Car();
          System.out.println(Car.numberOfCars + “ 대의 차가 생
   산되었습니다.” );
          Car hudson = new Car();
          Car marter = new Car();
          System.out.println(Car.numberOfCars + “ 대의 차가 생
   산되었습니다.” );
                                                          }
}
```

“static” 키워드

- “static” 메서드
  - 객체가 아닌 클래스의 행동을 나타내는 메서드로 객체(Object)로 호출되지 않음
  - 내부적으로 해당 객체의 “this”가 생성되지 않음
  - 반드시 static 메소드 안에서는 static 변수만 다뤄야 한다.

```
class Car {
   String name;
   int speed;
   static int numberOfCars;

   public Car() {
          …
          numberOfCars++;
   }
   …
   public static void printNumCars() {
          System.out.println(numberOfCars);
   }

   public static void main()
   {
          Car mcqueen = new Car();
          Car marter = new Car();

          marter.setSpeed(80);
          printNumCars();
     …
```

스택과 힙 – 힙(Heap)

- 힙(Heap)
  - 객체를 저장하기 위한 메모리 → 하나의 자바 프로그램마다 하나의 힙 생성
  - 객체가 new로 생성될 때 실제 객체가 생성되는 공간 → 모든 객체와 그 정보는 힙(Heap) 상에 거주함
    - 예: 모든 배열, 멤버 변수, 클래스 변수 (Static Variable)

- vs static
  - 자바 프로그램이 실행되기 전에 메모리에 확보되는 것이 아니라, 실행 중에 확보된다. 즉 동적 할당된다.
  - heap에 할당된 데이터는 가비지 컬렉션에 의해 관리된다.
  - class를 3번 new하면 3개의 heap이 할당된다. 즉 static과는 다르게 여러번 할당할 수 있다.

스택과 힙 – 힙(Heap)

- 객체의 삶과 죽음
  - 객체의 생성: 오직 "new"에 의해서만 생성
    - **`int[] intArray` *vs.* `int[] intArray = new int[5];`**
    - 자신을 참조하고 이는 변수가 하나 이상 있을 때만 살아 있음
  - 객체의 죽음: 자신의 유일한 참조 변수가(에)

    1)메쏘드가 종료로 Stack Frame이 사라질 때

    2)다른 객체가 대입되었을 때

    3)직접 null이 설정되었을 때

```
Duck d = new Duck(); //생성
d = new Duck(); //이전 객체 소멸
d = null; //객체 소멸
```

스택과 힙 – 스택(Stack)

- 스택 (Stack)
  - 메쏘드에 대한 정보를 저장하기 위한 메모리
    - 매개변수를 포함한 모든 지역변수 값
    - 수행되고 있는 메쏘드의 라인 정보
  - 메쏘드가 호출될 때마다, Stack의 상단에 새로운 스택 프레임(Stack Frame) 할당
    - 가장 늦게 호출된 메쏘드가 가장 상단에 올려짐 (Pushed)
    - 현재 수행 중인 메쏘드는 가장 상단의 스택 프레임(Stack Frame) 상의 메쏘드

스택과 힙 – 스택(Stack)

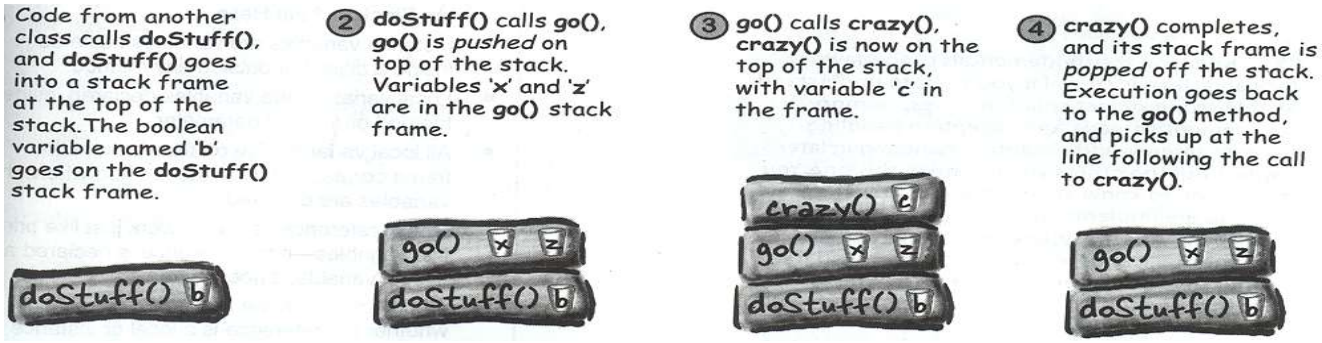
- 지역변수
  - 기본 자료형(Primitive Type) vs. 참조 자료형 (Reference Type)
  - 메쏘드 내부에서 정의되고 내부에서만 보임
  - 자신이 속한 메쏘드의 Stack Frame이 스택 메모리에 존재할 때만 생존

- stack 메모리는 가장 짧은 생명주기를 가지는 메모리 공간이다.
- 메서드가 호출될 때 잠깐 할당되었다고, 메서드가 종료될 때 사라지기 때문에 관리가 필요 없다.

스택과 힙 – 스택(Stack) 시나리오

```
…
   public void doStuff() {

          boolean b = true;
          go(4);
   }
   public void go(int x) {
          int z = x + 24;
          crazy()
   }
   public void crazy() {
          char c = ‘a’ ;
   }
…
```



가비지 컬렉션(Garbage Collection)

- 가비지 컬렉션 (Garbage collection)
  - 힙(Heap)에서 더 이상 필요치 않은 메모리를 점검하여 해제 시킴
  - JVM이 객체를 알아서 제거하는 기능을 자동적으로 수행함
  - JVM의 구현에 따라 환경에 맞춰 다양하게 반응
  - 주기적인 지연 발생

```
public class GarbageCollection {
  public static void main(String[] args) {
        String s;
        for( int i=0; i<10000; i++ ) {
              s = new String();
        }
        System.gc();
  }
}
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare