시험과목	Advanced Core Java
이름	

문제1. 반복문을 이용하여 369게임에서 박수를 쳐야 하는 경우의 수를 순서대로 화면에 출력해보세요.

실행 결과:

3 짝
6 짝
9 짝
13 짝
16 짝
19 짝
23 짝
26 짝
29 짝
30 짝
31 짝
32 짝
33 짝짝

문제 설명 :

1부터 99까지만 실행하세요.

문제 힌트 및 제한 :

1.각 수를 문자열로 다루어야 합니다. 정수를 String으로 바꾸기 위해 다음코드를 사용합니다.

```
String number = String.valueOf( 369 );
```

2.바꾼 String의 길이를 구해서 loop를 돌아야 합니다. 길이는 다음과 같이 구합니다.

```
int length = number.length();
```

3.loop 에서 각각의 위치의 문자(숫자)가 '3', '6', '9' 인지 확인하고 카운트를 셉니다. 이를 위해 다음 코드를 사용합니다.

```
char c = number.charAt( i );
```

문제2. 아래와 같은 출력이 나오도록 Money Class를 완성하세요.
단 MoneyTest Class의 main 메쏘드는 수정하지 마시오.

```
Money Class 구현을 완료 하였습니다.
```

문제 설명 :
제시된 소스 코드에 주석으로 된 부분을 채워서 프로그램을 완성합니다.

문제 힌트 및 제한 :
1.사칙 연산 메쏘드(add, minus, multiply, devide)에서는 자신의 금액(amount)과 인자로 넘어온 Money 객체의 금액(amount)을 계산하고, 계산된 금액으로 새로운 Money 객체로 생성하여 리턴한다.

2.equals 메쏘드에서는 인자로 넘어온 Object 객체가 Money 타입인지를 확인하고, Money 타입인 경우에 금액이 동일한지를 확인한다. 이 두가지 조건이 만족하는 경우에 true를 리턴한다.

문제3.Soundable 인터페이스의 sound() 메소드는 객체의 소리를 반환합니다.

Soudable.java

```
public interface Soundable {
   public String sound();
}
```

SoundTest.java

```
public class SoundTest {
  public static void main( String[] args ) {
     printSound( new Cat() );
     printSound( new Dog() );
     printSound( new Sparrow() );
     printSound( new Duck() );
  }
}
```

실행결과

"야옹"
"멍멍"
"짹짹"
"꽥꽥"

문제 힌트 및 제한 :

문제4. MainApp.java와 Stack.java는 다음과 같이 주어집니다. Main 클래스를 실행하였을 때 아래와 같은 결과가 출력되도록 MyStack 클래스를 작성하세요.

```
[출력]
.
Java
!!!
World
Hello
==================================
Hello
null
```

문제 설명 :

MyStack.java 를 완성합니다.

문제 힌트 및 제한 :

1 Main.java , Stack.java 파일은 수정하지 않습니다.

2 java.util 패키지의 클래스는 사용할 수 없습니다.

문제5. 버블정렬을 적용하는 자바코드를 완성하세요.
문제에 주어진 배열 [5, 9, 3, 8, 60, 20, 1] 를 내림차순으로 정렬하여 다음과 같은 출력이 되도록 완성하는 문제입니다.

```
<< 출력 결과 >>

Before sort.
5 9 3 8 60 20 1
After Sort.
60 20 9 8 5 3 1.
```

문제 설명 :

Sort.java 를 완성합니다.

문제 힌트 및 제한 :

버블정렬은 다음과 같습니다.
예를 들어, 초기값
[5, 9, 3, 8, 60, 20, 1]

1회
[9, 5, 3, 8, 60, 20, 1] -> 5, 9를 비교해서 뒤가 크므로 바꾼다
[9, 5, 3, 8, 60, 20, 1] -> 5, 3을 비교해서 뒤가 작으므로 제자리
[9, 5, 8, 3, 60, 20, 1] -> 3, 8를 비교해서 뒤가 크므로 바꾼다
[9, 5, 8, 60, 3, 20, 1] -> 3, 60을 비교해서 뒤가 크므로 바꾼다
[9, 5, 8, 60, 20, 3, 1] -> 3, 20을 비교해서 뒤가 크므로 바꾼다
[9, 5, 8, 60, 20, 3, 1] -> 3, 1을 비교해서 뒤가 작으므로 제자리

2회 : 같은 방식
[9, 8, 5, 60, 20, 3, 1]
[9, 8, 60, 5, 20, 3, 1]
[9, 8, 60, 5, 20, 3, 1]
[9, 8, 60, 20, 5, 3, 1]

3회 : 같은 방식, 마지막 결과
[9, 60, 20, 8, 5, 3, 1]

4회 : 같은 방식, 마지막 결과

[60, 20, 9, 8, 5, 3, 1]