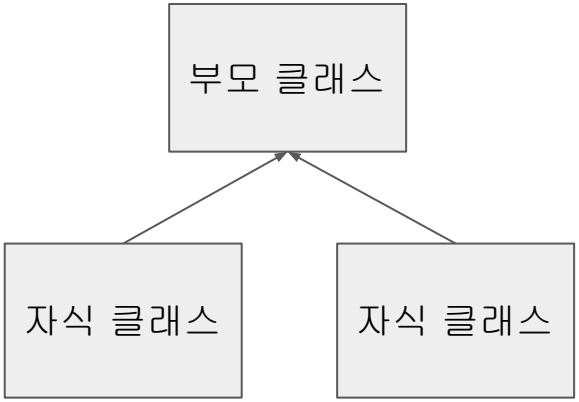
spring_한스푼(3주차)

2주차_review / 인터페이스
/ 기본클래스와 컬렉션 / 예외 처리

상속??

- 코드상으로 **<u>가리켜서 상속 받는 것</u>**.

- 예약어 : extends
- 사용법 : `public class 클래스명 extends 부모클래스 {}`
- 종류 :



상속 다형성_업캐스팅

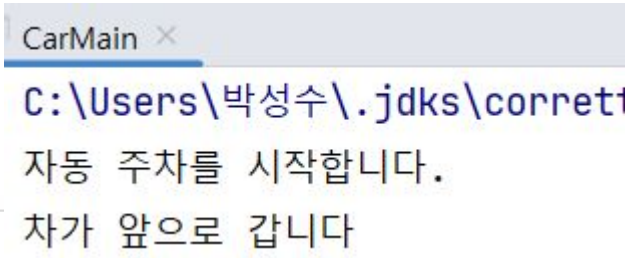
자식 클래스의 객체가 부모 클래스 타입으로 형변환 되는 것을 말한다.

저번 시간에 상속 Car(부모클래스)와 Kia(자식클래스)를 활용해서 main에 써보면..

Car car = new Kia;

부모 클래스로 객체를 선언 해지만 만들어진 객체는 Kia이다.

```
public class CarMain {
    public static void main(String[] args) {
        Car car = new Kia();
        car.parking();
        car.driver();
    }
}
```



다운 캐스팅

업캐스팅된 것을 다시 원상태로 돌리는 것을 말한다.

```
public class CarMain {
    public static void main(String[] args) {
        Kia car = new Kia();
        car.parking();
        car.driver();

        Kia kia = car;
        kia.parking();
        kia.driver();
        car.parking();
    }
}
```

```
CarMain
C:\Users\박성수\.jdks\corretto-
자동 주차를 시작합니다.
차가 앞으로 갑니다
자동 주차를 시작합니다.
차가 앞으로 갑니다
자동 주차를 시작합니다.
```

추상 클래스

- ‘구체적이지 않은 클래스’로 구현 코드가 없고 선언만 있는 메소드가 있는 클래스

- 예약어 : abstract

- 사용법

  public abstract class 클래스명{ }

추상 메소드

- 추상 클래스 안에 선언만 되어 있는 메소드
- 예약어 : abstract
- 사용법
  - public abstract void 메소드명(매개변수);

- 추상 클래스를 상속받은 자식클래스는 추상메소드를 재정의를 하여 구현을 필수로 해야한다.

같이 해보기

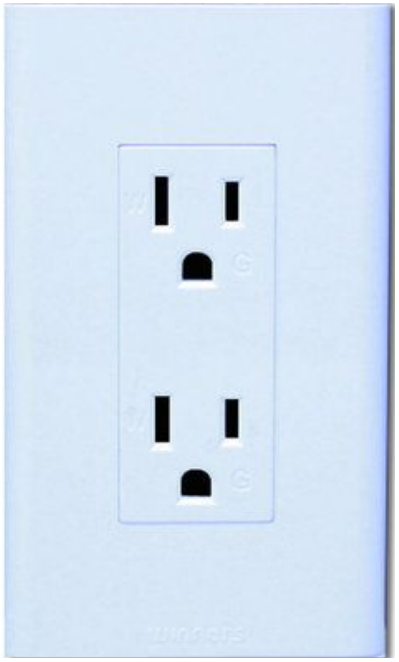
휴대폰의 기능들을 먼저 정해보자!

(전화 받기, 전화 끊기, 메세지 보내기, 전원 끄기, 전원 켜기 등등)

SamsungPhone 클래스를 만들어 상속 받아 기능들을 구현해보자

메인에서 실행을 해보자

미리보기 (인터페이스)



인터페이스(interface)

- public interface 클래스명{ } or 클래스 만들기 -> interface (사용법)
- 클래스에서 메소드를 구현하도록 강제 할 수 있는 기능.
- 모든 메서드가 추상 메서드로 이루어진 클래스로 형식적인 선언만 있고 구현은 없음.
- 인터페이스에 선언된 모든 메소드는 public abstract로 추상 메소드.
- 인터페이스에 선언된 모든 변수는 public static final로 상수

인터페이스 특징

- 상수 – 모든 변수는 상수로 변환 됨. public만 허용, public static final 생략
- 추상 메소드 – public abstract 생략 가능
- default 메소드
  - 인터페이스에 코드가 작성된 메소드
  - 인터페이스를 구현하는 클래스에 자동 상속. 구현 클래스에서 재정의 가능
  - public 접근 지정만 허용. 생략 가능
- private 메소드
  - 인터페이스 내에 메소드 코드가 작성되어야 함
  - 인터페이스 내에 있는 다른 메소드에 의해서만 호출 가능
- static 메소드 – public, private 모두 지정 가능. 생략하면 public

- 인터페이스의 객체 생성 불가
- 인터페이스 타입의 레퍼런스 변수 선언 가능(ex ExInterface Ex;)

같이 해보기!

인터페이스 Calc를 만들어보자!

인터페이스 상속

- 인터페이스 간에 상속 가능
  - 인터페이스를 상속하여 확장된 인터페이스 작성 가능
  - extends 키워드로 상속 선언

```
interface MobilePhoneInterface extends PhoneInterface {
   void sendSMS();          // 추상 메소드 추가
   void receiveSMS();       // 추상 메소드 추가
}
```

  - 인터페이스는 다중 상속 허용

```
interface MusicPhoneInterface extends PhoneInterface,
MP3Interface {
                .....
}
```

인터페이스 구현

- 인터페이스의 추상 메소드를 모두 구현한 클래스 작성
  - implements 키워드 사용
  - 여러 개의 인터페이스 동시 구현 가능

예시)

```
class SamsungPhone implements PhoneInterface { // 인터페이스 구현
   // PhoneInterface의 모든 메소드 구현
   public void sendCall() { System.out.println("띠리리리링"); }
   public void receiveCall() { System.out.println("전화가 왔습니다."); }

   // 메소드 추가 작성
   public void flash() { System.out.println("전화기에 불이 켜졌습니다."); }
}
```

  - 인터페이스를 구현한 클래스들을 하나의 인터페이스 타입으로 다룰 수 있음
  - 기능(메소드)의 구현을 강제함으로써, 클래스의 설계 또는 표준화를 유도 할 수 있음

같이 해보기!

인터페이스 Calc를 상속받아 구현해보자

직접 해보기!

게임 캐릭터의 기능을 구현해보자!

인터페이스 Player - jump(), run(), turn(), showLevelMessage()

클래스 - BeginnerLevel, AdvancedLevel, SuperLevel

  ㄴ>맴버 함수 - 인터페이스 받아 재정의 해보자!

미리보기(기본클래스)

기본클래스

- java.lang 패키지
  - 프로그래밍시 import하지 않아도 자동으로 추가됨(import java.lang.*;)
  - 많이 사용하는 기본 클래스들이 속한 패키지(String, Interger, System)

- Object 클래스 : 모든 클래스의 최상위 클래스
  - 모든 클래스는 Object 클래스에서 상속 받음
  - 모든 클래스는 Object 클래스의 메소드를 사용할 수 있음
  - 모든 클래스는 Object 클래스의 메소드 중 일부는 재정의 할 수 있음(final로 선언된 메소드는 재정의 할 수 없음)

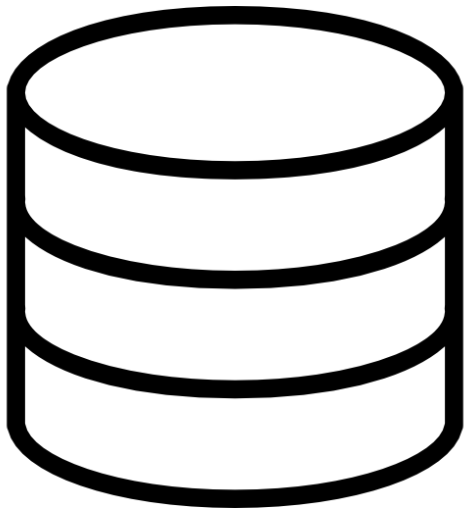
같이 해보기!

main에서 여러개의 메소드를 써보자!

- string 변수에서 여러개의 메소드를 써보자!

- List 변수에서 여러개의 메서드를 써보자!

미리보기(컬렉션)



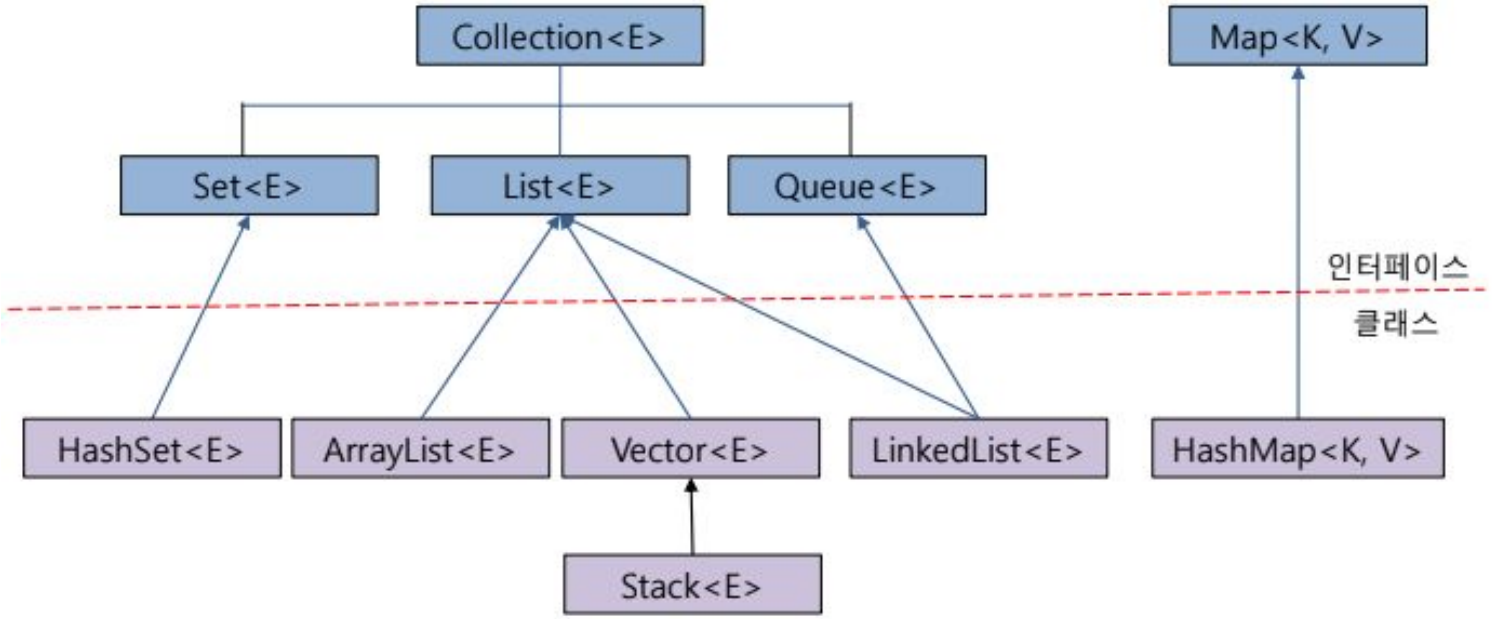
컬렉션(collection)

- 요소(element)라고 불리는 가변 개수의 객체들의 저장소
  - 객체들의 컨테이너라고도 불림
  - 요소의 개수에 따라 크기 자동 조절
  - 요소의 삽입, 삭제에 따른 요소의 위치 자동 이동

- 고정 크기의 배열을 다루는 어려움 해소함

- 다양한 객체들의 삽입, 삭제, 검색 등의 관리 용이

컬렉션 자바 인터페이스와 클래스



컬렉션의 특징

- 제네릭(generics) 기법으로 구현
  - 제네릭
    - ➔ 특정 타입만 다루지 않고, 여러 종류의 타입으로 변신할 수 있도록 클래스나 메소드를 일반화시키는 기법
    - ➔ 클래스나 인터페이스 이름에 <E>,<K>,<V> 등 타입 매개변수 포함
  - 제네릭 컬렉션 예시 : 벡터 Vector<E>
    - ➔ <E>에서 E에 구체적인 타입을 주어 구체적인 타입만 다루는 벡터로 활용
    - ➔ 정수만 다루는 컬렉션 벡터 Vector<Integer>
    - ➔ 문자열만 다루는 컬렉션 벡터 Vector<String>
- 컬렉션의 요소는 객체만 가능
  - int, char, double 등의 기본 타입으로 구체화 불가

제네릭(Generic)

- 클래스나 메소드를 형판에서 찍어내듯이 생산할 수 있도록 일반화된 형판을 만드는 기법
- 모든 종류의 데이터 타입을 다룰 수 있도록 일반화된 타입 매개 변수로 클래스(인터페이스)나 메소드를 작성하는 기법

같이 해보기!

도서 관리 프로그램을 만들어보자!

도서

마법 천자문 1,2,3,4,5 (만화책 분류)

컴활자격증 기출, 네관자격증 기출, 정처산기자격증 기출(자격증 분류)

리스트화 해서 출력해보기

미리보기(예외 처리)

오류란?

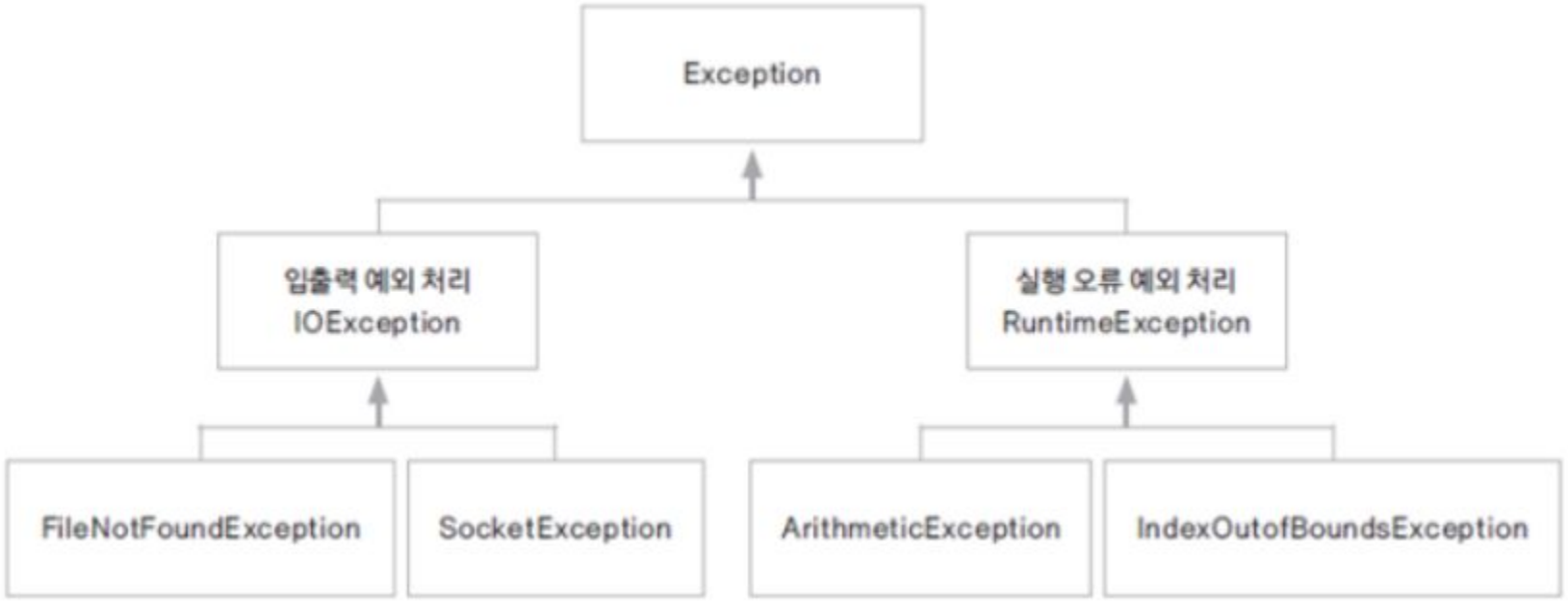
- 컴파일 오류 : 프로그램 코드 작성 중 발생하는 문법적 오류
- 실행 오류
  - ➔ 실행 중인 프로그램이 의도 하지 않은 동작을 하거나(bug) 프로그램이 중지되는 오류
  - ➔ 실행 오류 시 비정상 종료는 서비스 운영에 치명적이다.

- 오류가 발생한 경우 로그(log)를 남겨 추후 이를 분석하여 원인을 찾아야함.
- 예외 처리를 통해 프로그램의 비정상 종료를 막고 log를 남길 수 있음

오류와 예외 클래스

- 시스템 오류(error)
  - 가상 머신에서 발생
  - 프로그램에서 제어할 수 없음 (ex : 사용 가능한 동적 메모리 없는 경우, 스택메모리의 오버 플로우)
- 예외
  - 프로그램에서 제어 할 수 있음
    - ➔ 프로그램에서 파일을 읽어 사용하려는데 파일이 없는 경우
    - ➔ 네트워크로 데이터를 전송하려는 데 연결이 안 된 경우
    - ➔ 배열 값을 출력하려는 데 배열요소가 없는 경우

예외 클래스의 종류


Exception
입출력 예외 처리
IOException
실행 오류 예외 처리
RuntimeException
FileNotFoundException
SocketException
ArithmeticException
IndexOutofBoundsException

예외 처리하기(try-catch문)

- 예외가 발생할 때 발생 이후 처리하는 것이 try-catch문.

사용법

```
public static void main(String[] args) {
    int[] arr = new int[5];
    try {
        for (int i = 0; i <= 5; i++) {
            arr[i] = i;
            System.out.println(arr[i]);
        }
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println(e);
        System.out.println("예외 처리 부분");
    }
}
```

```
rretto-17.0.8.1\bin\java.exe "-javaagent:C:\Program F
0
1
2
3
4
java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 5
예외 처리 부분
```

같이 해보기!

숫자 아무거나 와 0을 나누면 예외가 나온다! 예외를 처리해보자

파일 입출력하면 예외가 발생할 수 있다. 예외를 처리해보자