SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

Java

SAMSUNG SW ACADEMY FOR YOUTH

Programming Language

객체지향 프로그래밍

- 추상클래스

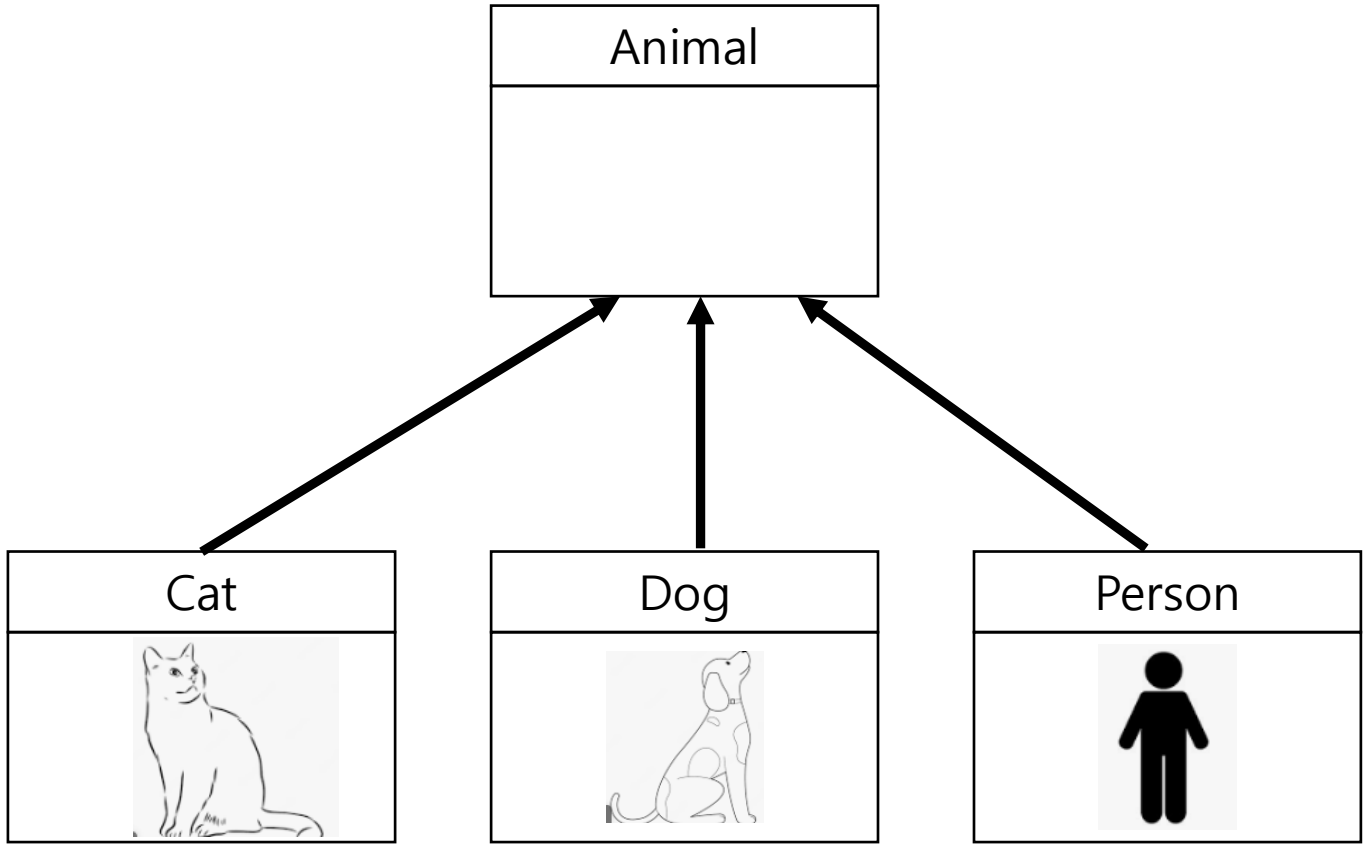
SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

추상클래스
(abstract class)

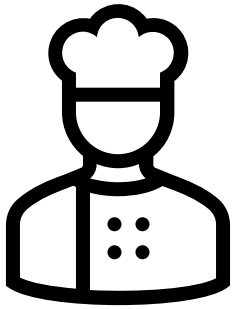
추상클래스 (abstract class)

- 생각해 봅시다.

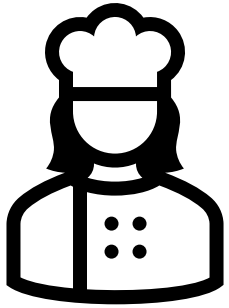


추상클래스 (abstract class)

추상 클래스 정의



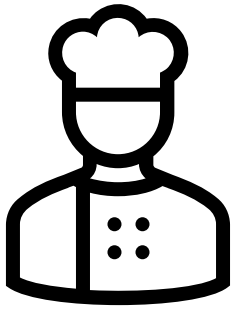
```
public class KFoodChef {
    String name;
    int age;
    String speciality;

    public void eat() {
        System.out.println("음식을 먹는다.");
    }

    public void cook() {
        System.out.println("한식을 요리한다.");
    }
}
```



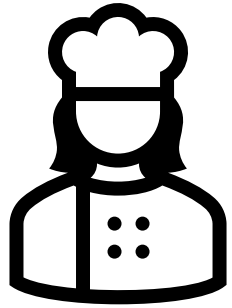
```
public class JFoodChef {
    String name;
    int age;
    String speciality;

    public void eat() {
        System.out.println("음식을 먹는다.");
    }

    public void cook() {
        System.out.println("일식을 요리한다.");
    }
}
```

추상클래스 (abstract class)

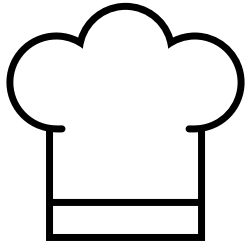
추상 클래스 정의



```
public class KFoodChef extends Chef{
    @Override
    public void cook() {
        System.out.println("한식을 요리한다.");
    }
}
```



```
public class JFoodChef extends Chef{
    @Override
    public void cook() {
        System.out.println("일식을 요리한다.");
    }
}
```



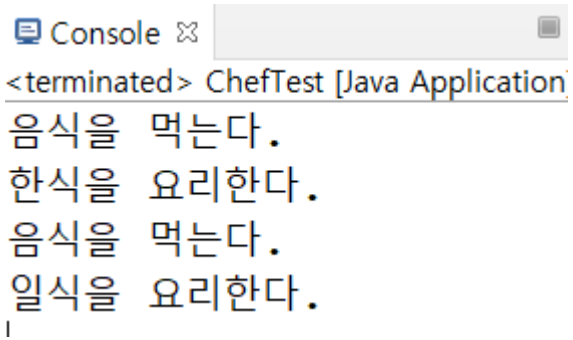
```
public class Chef {
    String name;
    int age;
    String speciality;

    public void eat() {
        System.out.println("음식을 먹는다.");
    }

    public void cook() {
        System.out.println("전공에 맞는 요리한다.");
    }
}
```

추상클래스 (abstract class)

추상 클래스 정의

```
public class Chef {
    String name;
    int age;
    String speciality;

    public void eat() {
        System.out.println("음식을 먹는다.");
    }

    public void cook() {
        System.out.println("전공에 맞는 요리한다.");
    }
}

public class KFoodChef extends Chef{
    @Override
    public void cook() {
        System.out.println("한식을 요리한다.");
    }
}

public class JFoodChef extends Chef{
    @Override
    public void cook() {
        System.out.println("일식을 요리한다.");
    }
}
```

```
public class ChefTest {
    public static void main(String[] args) {
        Chef[] chefs = new Chef[2];

        chefs[0] = new KFoodChef();
        chefs[1] = new JFoodChef();

        for(Chef chef : chefs) {
            chef.eat();
            chef.cook();
        }
    }
}
```



추상클래스 (abstract class)

추상 클래스 정의

- 한식요리사, 일식요리사 모두 cook 메서드를 가지고 있음.
- 조상 클래스 Chef에 선언하고 각 자손 클래스에서 override 예정
- 사용되지 않는 Chef 클래스에 cook() 메서드가 필요한가??

```
public class Chef {
    String name;
    int age;
    String speciality;

    public void eat() {
        System.out.println("음식을 먹는다.");
    }

    public void cook() {
        System.out.println("전공에 맞는 요리한다.");
    }
}
```

쓰이지 않는 코드를 지운다면?

```
for(Chef chef : chefs) {
    chef.eat();
    //Chef의 cook 메서드를 제거 했을 때.
    if (chef instanceof KFoodChef) {
        KFoodChef k = (KFoodChef)chef;
        k.cook();
    } else if (chef instanceof JFoodChef) {
        ((JFoodChef) chef).cook();
    }
}
```

```
Console
<terminated> ChefTest [Java Application]
음식을 먹는다.
한식을 요리한다.
음식을 먹는다.
일식을 요리한다.
```

추상클래스 (abstract class)

추상 클래스 정의

- cook() 메서드는 자손 클래스에서 반드시 재정의해서 사용되기 때문에 조상의 구현이 무의미
- 메서드의 선언부만 남기고 구현부는 ;(세미콜론)으로 대체
- 구현부가 없으므로 abstract 키워드를 메서드 선언부에 추가
- 객체를 생성할 수 없는 클래스라는 의미로 클래스 선언부에 abstract를 추가

```
public abstract class Chef {
    String name;
    int age;
    String speciality;

    public void eat() {
        System.out.println("음식을 먹는다.");
    }

    public abstract void cook();
}
```

추상클래스 (abstract class)

추상 클래스 특징

- abstract 클래스는 상속 전용의 클래스
- 클래스에 구현부가 없는 메서드가 있으므로 객체를 생성할 수 없음
- 상위 클래스 타입으로 자식을 참조할 수는 있음

```
//생성할 수 없음
Chef chef1 = new Chef();
//참조는 문제없음
Chef chef2 = new KFoodChef();
```

- 조상 클래스에서 상속 받은 abstract 메서드를 재정의 하지 않은 경우
  클래스 내부에 abstract 메서드가 있으므로 자식 클래스는 abstract 클래스가 되어야함.

추상 클래스 사용하는 이유

- abstract 클래스는 구현의 강제를 통해 프로그램의 안정성 향상

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

다음 방송에서 만나요!

삼성 청년 SW 아카데미