KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

제네릭

[KB] IT's Your Life

KB국민은행

제네릭

- 다음 Box 클래스의 content 멤버를 제너릭 타입으로 정의하세요.

```
package ch13.sec01;

public class Box_________ {
  public _______ content;
}
```

Box.java

```
package ch13.sec01;

public class Box<T> {
  public T content;
}
```

앞에서 정의한 Box 클래스를 이용하여 다음 코드를 완성하세요.

```
package ch13.sec01;

public class GenericExample {
  public static void main(String[] args) {

    _____________________________________________;
    box1.content = "안녕하세요.";
    String str = box1.content;
    System.out.println(str);

    _____________________________________________;
    box2.content = 100;
    int value = box2.content;
    System.out.println(value);
  }
}
```

GenericExample.java

```
package ch13.sec01;

public class GenericExample {
  public static void main(String[] args) {
    //Box<String> box1 = new Box<String>();
    Box<String> box1 = new Box<>();  // Box 생성시 타입파라미터 대신 String으로 대체
    box1.content = "안녕하세요.";
    String str = box1.content;
    System.out.println(str);

    //Box<Integer> box2 = new Box<Integer>();
    Box<Integer> box2 = new Box<>();  // Box 생성시 타입파라미터 대신 Integer로 대체
    box2.content = 100;
    int value = box2.content;
    System.out.println(value);
  }
}
```

```
안녕하세요.
100
```

제네릭

- **다음 조건을 만족하는 Product 클래스를 정의하세요.**
  - 멤버 변수 kind, model을 제너릭으로 운영
  - 각가의 멤버에 대해 Getter, Setter 메서드를 직접 정의(Lombok 사용 불가)

```
package ch13.sec02.exam01;

public class Product {

}
```

Product.java

```
package ch13.sec02.exam01;

//제네릭 타입
public class Product<K, M> {  // 타입 파라미터로 K와 M 정의
  //타입 파라미터를 필드 타입으로 사용
  private K kind;
  private M model;

  //타입 파라미터를 리턴 타입과 매개 변수 타입으로 사용
  public K getKind() { return this.kind; }
  public M getModel() { return this.model; }
  public void setKind(K kind) { this.kind = kind; }
  public void setModel(M model) { this.model = model; }
}
```

다음 더미 클래스를 정의하세요.

```
package ch13.sec02.exam01;

public class Tv {
}
```

```
package ch13.sec02.exam01;

public class Car {
}
```

Product, Tv, Car 등의 클래스를 이용하여 아래 코드를 완성하세요.

```
package ch13.sec02.exam01;

public class GenericExample {
  public static void main(String[] args) {
    _____________________ product1 = _____________________;
    product1.setKind(new Tv());
    product1.setModel("스마트Tv");

    ________ tv = product1.getKind();
    ________ tvModel = product1.getModel();

    _____________________ product2 = _____________________;
    product2.setKind(new Car());
    product2.setModel("SUV자동차");

    _______ car = product2.getKind();
    _______ carModel = product2.getModel();
  }
}
```

GenericExample.java

```
package ch13.sec02.exam01;

public class GenericExample {
  public static void main(String[] args) {
    //K는 Tv로 대체, M은 String으로 대체
    Product<Tv, String> product1 = new Product<>();

    //Setter 매개값은 반드시 Tv와 String을 제공
    product1.setKind(new Tv());
    product1.setModel("스마트Tv");

    //Getter 리턴값은 Tv와 String이 됨
    Tv tv = product1.getKind();
    String tvModel = product1.getModel();

    //K는 Car로 대체, M은 String으로 대체
    Product<Car, String> product2 = new Product<>();

    //Setter 매개값은 반드시 Car와 String을 제공
    product2.setKind(new Car());
    product2.setModel("SUV자동차");

    //Getter 리턴값은 Car와 String이 됨
    Car car = product2.getKind();
    String carModel = product2.getModel();
  }
}
```

다음 인터페이스와 클래스를 정의하세요.

```
package ch13.sec02.exam02;

public interface Rentable<P> {
  P rent();
}
```

```
package ch13.sec02.exam02;

public class Home {
  public void turnOnLight() {
    System.out.println("전등을 켭니다.");
  }
}
```

```
package ch13.sec02.exam02;

public class Car {
  public void run() {
    System.out.println("자동차가 달립니다.");
  }
}
```

Rentable 인터페이스를 구현하는 HomeAgency, CarAgency 클래스를 정의하세요.

- HomeAgency의 rent()는 Home 클래스 인스턴스를 리턴
- CarAgency의 rent()는 Car 클래스 인스턴스를 리턴

```
package ch13.sec02.exam02;

public class HomeAgency {

}
```

```
package ch13.sec02.exam02;

public class CarAgency {

}
```

HomeAgency.java 인터페이스에서의 제너릭 사용

```
package ch13.sec02.exam02;

public class HomeAgency implements Rentable<Home> {  // 타입 파라미터 P를 Home으로 대체
  @Override
  public Home rent() { // 리턴 타입이 반드시 Home이어야 함
    return new Home();
  }
}
```

CarAgency.java

```
package ch13.sec02.exam02;

public class CarAgency implements Rentable<Car> {  // 타입 파라미터 P를 Car으로 대체
  @Override
  public Car rent() {// 리턴 타입이 반드시 Car여야 함
    return new Car();
  }
}
```

앞에서 정의한 인터페이스와 클래스를 이용하여 다음 출력이 나오도록 수정하세요.

```
package ch13.sec02.exam02;

public class GenericExample {
  public static void main(String[] args) {
    ____________________________________________________;
    __________ = homeAgency.rent();
    home.turnOnLight();

    ____________________________________________________;
    __________ = carAgency.rent();
    car.run();
  }
}
```

```
전등을 켭니다.
자동차가 달립니다.
```

GenericExample.java 인터페이스에서의 제너릭 사용

```
package ch13.sec02.exam02;

public class GenericExample {
  public static void main(String[] args) {
    HomeAgency homeAgency = new HomeAgency();
    Home home = homeAgency.rent();
    home.turnOnLight();

    CarAgency carAgency = new CarAgency();
    Car car = carAgency.rent();
    car.run();
  }
}
```

```
전등을 켭니다.
자동차가 달립니다.
```