단국대학교 컴퓨터공학 JAVA 프로그래밍 II 중간고사 (2021 Fall) 2021년 10월 18일 13:30-14:40

학과 _____ 학번 _____ 이름 _____ 비밀번호 _____

중간고사 실습문제

담당교수: 단국대학교 컴퓨터공학 박경신

- **아래의 3개 문제 중에서 1개를 선택하여 구현한다. 전체 프로젝트 실행 코드를 JAVA21-2-Midterm-학번-이름.zip로 만들어서 e-learning에 제출 (보고서는 코드에 주석으로)**

1. HW4 에 스트래터지 패턴(Strategy Pattern)을 사용한다. IPeriodicElementFinder 를 사용하여 find 알고리즘을 선택 및 교환해서 사용하는 프로그램을 작성한다. (100 점)

1.1 인터페이스 정의 및 구현 클래스 (70 점)

```
public interface IPeriodicElementFinder {
    void find(List<PeriodicElement> list);
}
```

NumberFinder class find PeriodicElement by Number.
NameFinder class find PeriodicElement by Name.
SymbolFinder class find PeriodicElement by Symbol.
PeriodFinder class find PeriodicElement by Period.
GroupFinder class find PeriodicElement by Group.
StateFinder class find PeriodicElement by State(gas/liq/solid/unknown).

1.2 인터페이스 사용 (10 점)

PeriodicElementFinder class find the list of PeriodicElement using IPeriodicElementFinder.

1.3 전체적인 구성 및 패턴 구조 설계 (10 점)

1.4 MainTest 클래스에 실행 부분 및 본인 추가 코드 (10 점)

학과 _____ 학번 _____ 이름 _____ 비밀번호 _____

2. HW2 에 데코레이터 패턴(Decorator Pattern)을 사용한다. ImageProcessDecorator 인 ImageGrayscale, ImageBlur, ImageNegative, ImageEdgeDetect, ImageRotate(본인이 작성한 것으로 사용), etc 로 꾸며주는 프로그램을 작성한다. Component 에 해당되는 객체 PlaneImage 는 일반이미지를 화면에 출력한다. PlaneImage 는 추상클래스인 ImageProcessor 를 상속받아서. 여기에 데코레이터인 ImageGrayscale 를 추가하면 현재 이미지를 grayscale 로 바꾼다. ImageBlur 를 추가하면 현재 이미지를 blur 하게 바꾼다. 여기에 ImageNegative 가 추가되면 현재 이미지를 negative 하게 바꾼다. 여기에 ImageEdgeDetect 를 추가하면 현재 이미지를 edge detect 하게 바꾼다. (50 점)

2.1 Component 인터페이스 정의 및 실제 클래스 구현 (20점)

```
public abstract class ImageProcessor {
    protected BufferedImage image = null;
    protected String name = null; // filename only
    protected String ext = null; // extension only
    protected String path = null; // path only
    public void load(String filename) {
        // 내부구현
    }
    public void process() {
        // decorated process
        // save image
    }
    public abstract BufferedImage process(BufferedImage image); //
process
    public abstract String name(); // name
}
public class PlaneImage extends ImageProcessor {
    public PlaneImage(String filename) {
       load(filename); // load image
    }
   @Override
   public String name() {
       return name; // name
   }
```

```
    @Override
    public BufferedImage process(BufferedImage image) {
        return image; // buffered image
    }
}
```

2.2 Decorator 인터페이스 정의 및 실제 클래스 구현(60점)

```
public abstract class ImageProcessDecorator extends ImageProcessor

public class ImageGrayscale extends ImageProcessDecorator

public class ImageBlur extends ImageProcessDecorator

public class ImageNegative extends ImageProcessDecorator

public class ImageEdgeDetect extends ImageProcessDecorator

public class ImageRotate extends ImageProcessDecorator (본인클래스 사용)
```

2.3 나머지 코드 (main()에서 기본 Component와 Decorator 사용 코드) (20점)

MainTest class 에서 C:/Temp/cat1.jpg 로딩해서, Grayscale + Blur + Negative

=> C:/Temp/cat1GrayscaleBlurNegative.jpg 이미지 저장

MainTest class 에서 C:/Temp/cat2.jpg 로딩해서, Rotate + EdgeDetect =>

C:/Temp/cat2RotateEdgeDetect.jpg 이미지 저장

3. HW3 에 추상 팩토리 패턴(Abstract Factory Pattern)을 사용한 AbstractAnimalFactory 를 작성한다.

3.1 Habitat 열거형 (LAND/SEA/WETLAND), AbstractAnimalFactory 클래스를 상속받는 LandAnimalFactory, SeaAnimalFactory, WetlandAnimalFactory 클래스 정의, AnimalFactoryProvider 클래스 정의 (30점)

```
public enum Habitat {
   LAND, SEA, WETLAND
}
public abstract class AbstractAnimalFactory {
   public abstract Animal create(AnimalType type);
}
public class LandAnimalFactory extends AbstractAnimalFactory {
   // LandMammal, LandBird, LandReptile, LandAmphibian 생성
}
public class SeaAnimalFactory extends AbstractAnimalFactory {
   // SeaMammal, SeaReptile, SeaFish 생성
}
public class WetlandAnimalFactory extends AbstractAnimalFactory {
   // WetlandMammal, WetlandBird, WetlandReptile, WetlandAmphibian,
   WetlandFish 생성
}
public class AnimalFactoryProvider {
   public static AbstractAnimalFactory getFactory(Habitat type) {
   // LandAnimalFactory, SeaAnimalFactory, WetlandAnimalFactory 생성
   }
}
```

3.3 Animal 클래스에 **Habitat 멤버필드 추가**, Mammal, Bird, Reptile, Amphibian, Fish **추상클래**

스로 전환 및 LandMammal, SeaMammal, WetlandMammal, LandBird, WetlandBird, SeaBird, LandReptile, SeaReptile, WetlandReptile, LandAmphibian, (SeaAmphibian없음) WetlandAmphibian, (LandFish없음), SeaFish, WetlandFish 클래스 정의 (40점)

```
public class LandMammal extends Mammal {
   public LandMammal () {
      this.move = "walk";
      this.habitat = Habitat.LAND;
   }
   @Override
   public String toString() {
      return "LandMammal [name=" + name + ", move=" + move + ", breath="
+ breath + ", reproduce=" + reproduce + ", numberOfLegs=" + numberOfLegs
+ ", habitat=" + habitat + ", type()=" + type() + "]";
   }
}
```

3.3 AnimalKingdom 클래스에 findAnimal, findObserver 메소드 추가 (10점)

```
public class AnimalKingdom {
   Animal findAnimal(String name) { // 내부구현 }
   Observer findObserver(String name) { // 내부구현 }
}
```

3.4 AnimalImport 클래스 정의와 MainTest 클래스의 main()에서 "Animals.csv"를 load해서 랜덤한 sleep후 Animal 추가/삭제 및 Observer 추가/삭제(20점)

AnimalImporter load "Animals.csv"에서 AnimalFactoryProvider와 AbstractAnimalFactory를 사용한 Animal 생성

-끝-