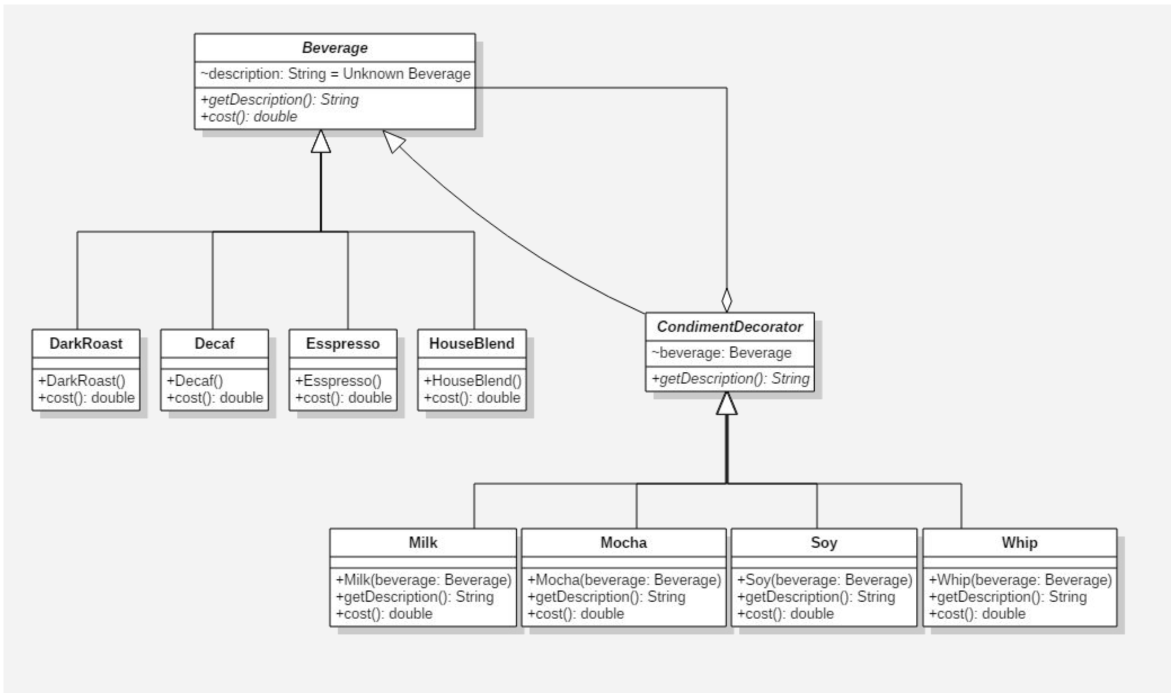
디자인패턴 실습 보고서

(8) Decorator Pattern

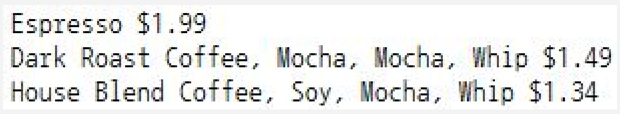
2011150 박동욱

1-1. Beverage - UML



1-2. Beverage - JAVA

실행화면



```
Espresso $1.99
Dark Roast Coffee, Mocha, Mocha, Whip $1.49
House Blend Coffee, Soy, Mocha, Whip $1.34
```

소스 코드

StarBuzzCoffee.java (main)

```java
public class StarbuzzCoffee {

	public static void main(String args[]) {
		Beverage beverage = new Espresso();
		System.out.println(beverage.getDescription()
				+ " $" + beverage.cost());

		Beverage beverage2 = new DarkRoast();
		beverage2 = new Mocha(beverage2);
		beverage2 = new Mocha(beverage2);
		beverage2 = new Whip(beverage2);
		System.out.println(beverage2.getDescription()
				+ " $" + beverage2.cost());

		Beverage beverage3 = new HouseBlend();
		beverage3 = new Soy(beverage3);
		beverage3 = new Mocha(beverage3);
		beverage3 = new Whip(beverage3);
```

```
                System.out.println(beverage3.getDescription()
                                + " $" + beverage3.cost());
        }
}
```

Beverage.java

```
public abstract class Beverage {
        String description = "Unknown Beverage";

        public String getDescription() {
                return description;
        }

        public abstract double cost();
}
```

CondimentDecorator.java

```
public abstract class CondimentDecorator extends Beverage {
        Beverage beverage;
        public abstract String getDescription();
}
```

DarkRoast.java

```
public class DarkRoast extends Beverage {
        public DarkRoast() {
                description = "Dark Roast Coffee";
        }

        public double cost() {
                return .99;
        }
}
```

Decaf.java

```
public class Decaf extends Beverage {
        public Decaf() {
                description = "Decaf Coffee";
        }

        public double cost() {
                return 1.05;
        }
}
```

Espresso.java

```
public class Espresso extends Beverage {

        public Espresso() {
                description = "Espresso";
        }

        public double cost() {
                return 1.99;
        }
}
```

HouseBlend.java

```java
public class HouseBlend extends Beverage {
	public HouseBlend() {
		description = "House Blend Coffee";
	}

	public double cost() {
		return .89;
	}
}
```

Milk.java

```java
public class Milk extends CondimentDecorator {

	public Milk(Beverage beverage) {
		this.beverage = beverage;
	}

	public String getDescription() {
		return beverage.getDescription() + ", Milk";
	}

	public double cost() {
		return .10 + beverage.cost();
	}
}
```

Mocha.java

```java
public class Mocha extends CondimentDecorator {

	public Mocha(Beverage beverage) {
		this.beverage = beverage;
	}

	public String getDescription() {
		return beverage.getDescription() + ", Mocha";
	}

	public double cost() {
		return .20 + beverage.cost();
	}
}
```

Soy.java

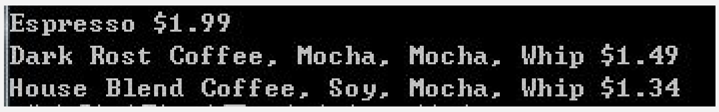
```java
public class Soy extends CondimentDecorator {

	public Soy(Beverage beverage) {
		this.beverage = beverage;
	}

	public String getDescription() {
		return beverage.getDescription() + ", Soy";
	}
	public double cost() {
		return .15 + beverage.cost();
	}
}
```

Whip.java

```
public class Whip extends CondimentDecorator {

	public Whip(Beverage beverage) {
		this.beverage = beverage;
	}

	public String getDescription() {
		return beverage.getDescription() + ", Whip";
	}

	public double cost() {
		return .10 + beverage.cost();
	}
}
```

1-3. Beverage - C++

실행화면



```
Espresso $1.99
Dark Rost Coffee, Mocha, Mocha, Whip $1.49
House Blend Coffee, Soy, Mocha, Whip $1.34
```

소스 코드

StarBuzzCoffee.cpp (main)

```
void main()
{
	Beverage *beverage = new Espresso();
	cout << beverage->getDescription() << " $" << beverage->cost() << endl;

	Beverage *beverage2 = new DarkRoast();
	beverage2 = new Mocha(beverage2);
	beverage2 = new Mocha(beverage2);
	beverage2 = new Whip(beverage2);
	cout << beverage2->getDescription() << " $" << beverage2->cost() << endl;

	Beverage *beverage3 = new HouseBlend();
	beverage3 = new Soy(beverage3);
	beverage3 = new Mocha(beverage3);
	beverage3 = new Whip(beverage3);
	cout << beverage3->getDescription() << " $" << beverage3->cost() << endl;

	delete beverage;
	delete beverage2;
	delete beverage3;
}
```

Beverage.h

```
class Beverage
{
protected:
	string description = "Unknown Beverage";
public:
	virtual string getDescription(){
		return description;
	}

	virtual double cost() = 0;
};
```

CondimentDecorator.h

```
class CondimentDecorator : public Beverage
{
protected:
	Beverage *beverage;
public:
	virtual string getDescription() = 0;
};
```

DarkRoast.h

```
class DarkRoast : public Beverage
{
public:
	DarkRoast(){
		description = "Dark Rost Coffee";
	}

	virtual double cost(){
		return .99;
	}
};
```

Decaf.h

```
class Decaf : public Beverage
{
public:
	Decaf(){
		description = "Decaf Coffee";
	}

	virtual double cost(){
		return 1.05;
	}
};
```

Espresso.h

```
class Espresso : public Beverage
{
public:
	Espresso(){
		description = "Espresso";
```

```
        }

        virtual double cost(){
                return 1.99;
        }
};
```

HouseBlend.h

```
class HouseBlend : public Beverage
{
public:
        HouseBlend(){
                description = "House Blend Coffee";
        }

        virtual double cost(){
                return .89;
        }
};
```

Milk.h

```
class Milk : public CondimentDecorator
{
public:
        Milk(Beverage *beverage){
                this->beverage = beverage;
        }

        virtual string getDescription(){
                return beverage->getDescription() + ", milk";
        }

        double cost(){
                return .10 + beverage->cost();
        }
};
```

Mocha.h

```
class Mocha : public CondimentDecorator
{
public:
        Mocha(Beverage *beverage){
                this->beverage = beverage;
        }

        virtual string getDescription(){
                return beverage->getDescription() + ", Mocha";
        }

        double cost(){
                return .20 + beverage->cost();
        }
};
```

Soy.h

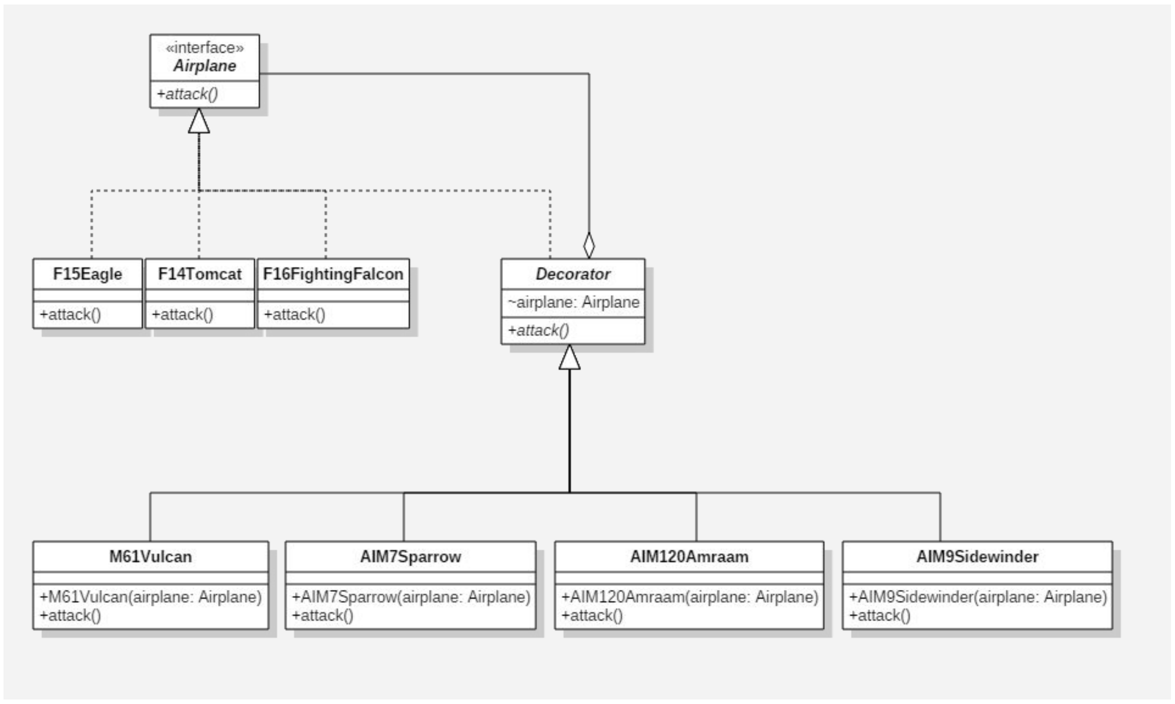
```
class Soy : public CondimentDecorator
```

```
{
public:
	Soy(Beverage *beverage){
		this->beverage = beverage;
	}

	virtual string getDescription(){
		return beverage->getDescription() + ", Soy";
	}

	double cost(){
		return .15 + beverage->cost();
	}
};
```

Whip.h

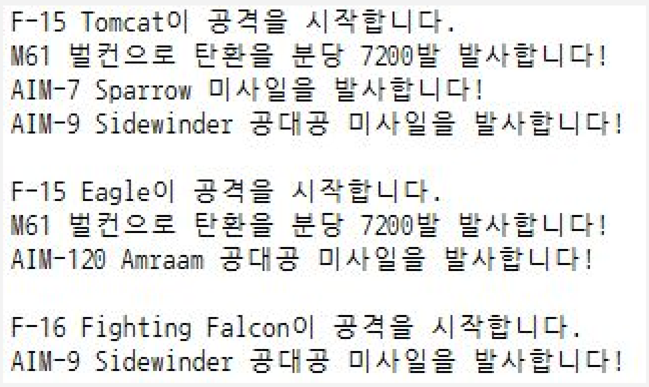
```
class Whip : public CondimentDecorator
{
public:
	Whip(Beverage *beverage){
		this->beverage = beverage;
	}

	virtual string getDescription(){
		return beverage->getDescription() + ", Whip";
	}

	double cost(){
		return .10 + beverage->cost();
	}
};
```

2-1. Airplane - UML



2-2. Airplane - JAVA

실행화면



```
F-15 Tomcat이 공격을 시작합니다.
M61 벌컨으로 탄환을 분당 7200발 발사합니다!
AIM-7 Sparrow 미사일을 발사합니다!
AIM-9 Sidewinder 공대공 미사일을 발사합니다!

F-15 Eagle이 공격을 시작합니다.
M61 벌컨으로 탄환을 분당 7200발 발사합니다!
AIM-120 Amraam 공대공 미사일을 발사합니다!

F-16 Fighting Falcon이 공격을 시작합니다.
AIM-9 Sidewinder 공대공 미사일을 발사합니다!
```

소스 코드

AirplaneTest.java (main)

```java
public class AirplaneTest {

	public static void main(String[] args) {
		Airplane airplane = new F14Tomcat();
		airplane = new M61Vulcan(airplane);
		airplane = new AIM7Sparrow(airplane);
		airplane = new AIM9Sidewinder(airplane);
		airplane.attack();
		System.out.println();

		Airplane airplane2 = new F15Eagle();
		airplane2 = new M61Vulcan(airplane2);
		airplane2 = new AIM120Amraam(airplane2);
		airplane2.attack();
		System.out.println();

		Airplane airplane3 = new F16FightingFalcon();
		airplane3 = new AIM9Sidewinder(airplane3);
		airplane3.attack();
	}

}
```

Airplane.java

```
public interface Airplane {
	public void attack();
}
```

Decorator.java

```
public abstract class Decorator implements Airplane {
	Airplane airplane;
	public abstract void attack();
}
```

F14Tomcat.java

```
public class F14Tomcat implements Airplane {

	@Override
	public void attack() {
		System.out.println("F-15 Tomcat이 공격을 시작합니다.");
	}
}
```

F15Eagle.java

```
public class F15Eagle implements Airplane {

	@Override
	public void attack() {
		System.out.println("F-15 Eagle이 공격을 시작합니다.");
	}
}
```

F16FightingFalcon.java

```
public class F16FightingFalcon implements Airplane {

	@Override
	public void attack() {
		System.out.println("F-16 Fighting Falcon이 공격을 시작합니다.");
	}
}
```

M61Vulcan.java

```java
public class M61Vulcan extends Decorator {

	public M61Vulcan(Airplane airplane) {
		this.airplane = airplane;
	}

	@Override
	public void attack() {
		airplane.attack();
		System.out.println("M61 벌컨으로 탄환을 분당 7200발 발사합니다!");
	}
}
```

AIM120Amraam.java

```java
public class AIM120Amraam extends Decorator {

	public AIM120Amraam(Airplane airplane) {
		this.airplane = airplane;
	}

	@Override
	public void attack() {
		airplane.attack();
		System.out.println("AIM-120 Amraam 공대공 미사일을 발사합니다!");
	}
}
```

AIM7Sparrow.java

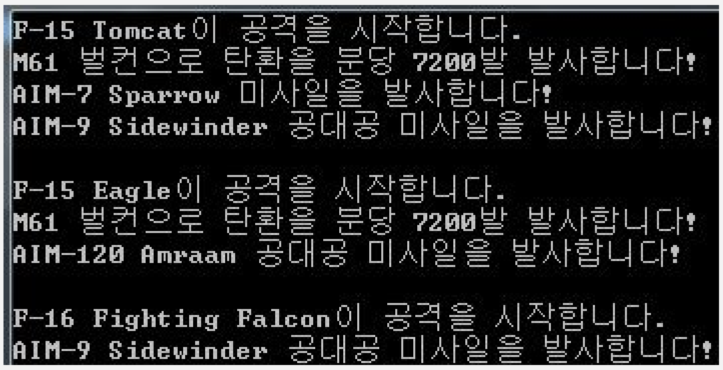
```java
public class AIM7Sparrow extends Decorator {

	public AIM7Sparrow(Airplane airplane) {
		this.airplane = airplane;
	}

	@Override
	public void attack() {
		airplane.attack();
		System.out.println("AIM-7 Sparrow 미사일을 발사합니다!");
	}
}
```

AIM9Sidewinder.java

```java
public class AIM9Sidewinder extends Decorator {

	public AIM9Sidewinder(Airplane airplane) {
		this.airplane = airplane;
	}

	@Override
	public void attack() {
		airplane.attack();
		System.out.println("AIM-9 Sidewinder 공대공 미사일을 발사합니다!");
	}
}
```

2-3. Airplane - C++

실행화면



```
F-15 Tomcat이 공격을 시작합니다.
M61 벌컨으로 탄환을 분당 7200발 발사합니다!
AIM-7 Sparrow 미사일을 발사합니다!
AIM-9 Sidewinder 공대공 미사일을 발사합니다!

F-15 Eagle이 공격을 시작합니다.
M61 벌컨으로 탄환을 분당 7200발 발사합니다!
AIM-120 Amraam 공대공 미사일을 발사합니다!

F-16 Fighting Falcon이 공격을 시작합니다.
AIM-9 Sidewinder 공대공 미사일을 발사합니다!
```

소스 코드

AirplaneTest.cpp (main)

```
void main()
{
	Airplane *airplane = new F14Tomcat();
	airplane = new M61Vulcan(airplane);
	airplane = new AIM7Sparrow(airplane);
	airplane = new AIM9Sidewinder(airplane);
	airplane->attack();
	cout << endl;

	Airplane *airplane2 = new F15Eagle();
	airplane2 = new M61Vulcan(airplane2);
	airplane2 = new AIM120Amraam(airplane2);
	airplane2->attack();
	cout << endl;

	Airplane *airplane3 = new F16FightingFalcon();
	airplane3 = new AIM9Sidewinder(airplane3);
	airplane3->attack();

	delete airplane;
	delete airplane2;
	delete airplane3;
}
```

Airplane.h

```
class Airplane
{
public:
	virtual void attack() = 0;
};
```

Decorator.h

```
class Decorator : public Airplane
```

```
{
protected:
        Airplane *airplane;
public:
        virtual void attack() = 0;
};
```

F14Tomcat.h

```
class F14Tomcat : public Airplane
{
public:
        virtual void attack(){
                cout << "F-15 Tomcat이 공격을 시작합니다." << endl;
        }
};
```

F15Eagle.h

```
class F15Eagle : public Airplane
{
public:
        virtual void attack(){
                cout << "F-15 Eagle이 공격을 시작합니다." << endl;
        }
};
```

F16FightingFalcon.h

```
class F16FightingFalcon : public Airplane
{
public:
        virtual void attack(){
                cout << "F-16 Fighting Falcon이 공격을 시작합니다." << endl;
        }
};
```

M61Vulcan.h

```
class M61Vulcan : public Decorator
{
public:
        M61Vulcan(Airplane *airplane){
                this->airplane = airplane;
        }
        virtual void attack(){
                airplane->attack();
                cout << "M61 벌컨으로 탄환을 분당 7200발 발사합니다!" << endl;
        }
};
```

AIM120Amraam.h

```
class AIM120Amraam : public Decorator
{
public:
        AIM120Amraam(Airplane *airplane){
                this->airplane = airplane;
        }
        virtual void attack(){
                airplane->attack();
                cout << "AIM-120 Amraam 공대공 미사일을 발사합니다!" << endl;
        }
};
```

AIM7Sparrow.h

```
class AIM7Sparrow : public Decorator
{
public:
        AIM7Sparrow(Airplane *airplane){
                this->airplane = airplane;
        }
        virtual void attack(){
                airplane->attack();
                cout << "AIM-7 Sparrow 미사일을 발사합니다!" << endl;
        }
};
```

AIM9Sidewinder.h

```
class AIM9Sidewinder : public Decorator
{
public:
        AIM9Sidewinder(Airplane *airplane){
                this->airplane = airplane;
        }
        virtual void attack(){
                airplane->attack();
                cout << "AIM-9 Sidewinder 공대공 미사일을 발사합니다!" << endl;
        }
};
```