Decorator Design Pattern.

What id decorator Design Pattern?

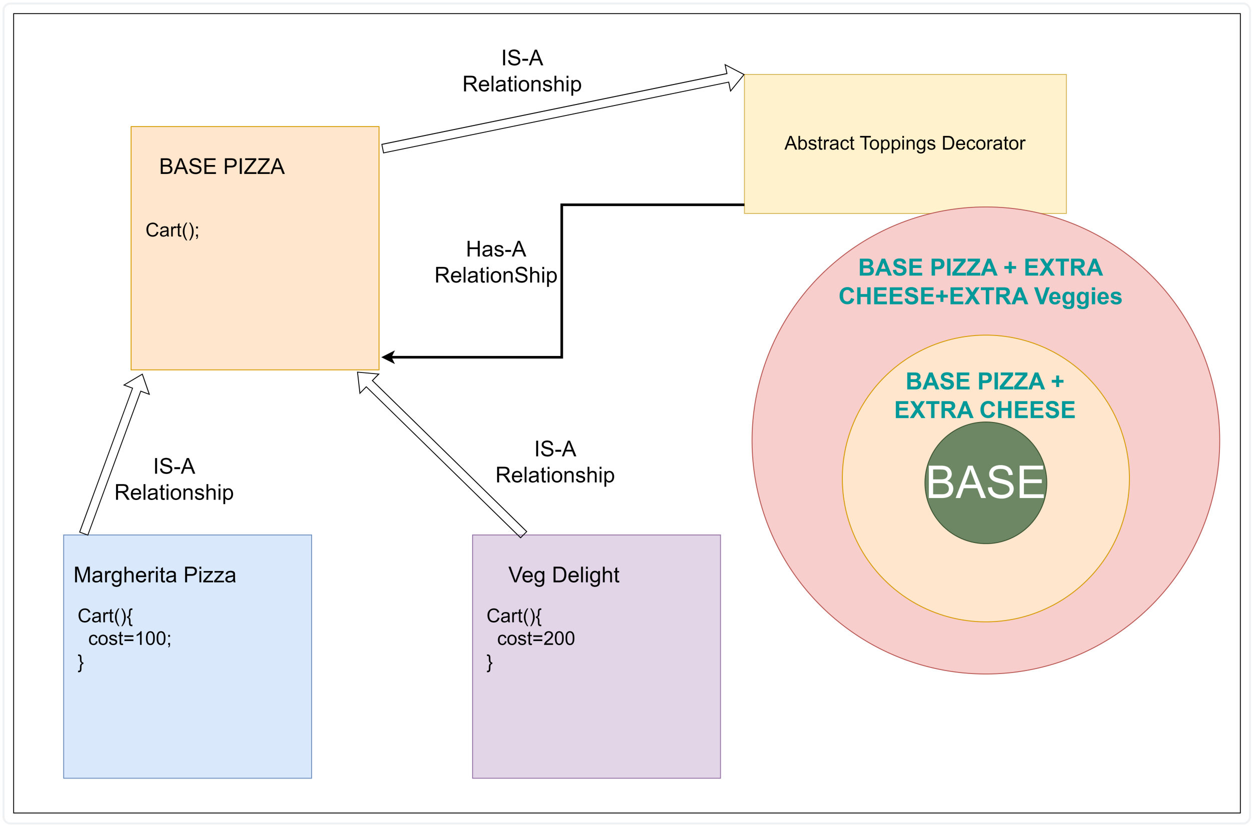
- Here we are decorating Object's by add some extra feature on existing object.

Examples:

- Base Pizza after we are adding veggies,cheese and jalpena.
- Base coffee, Expresso, extra milk,
- Base Car cover sheet, fog light

Why we need Decorator Pattern.

- ISSUE: Class Explosion.
- In example of car we cannot make classes for every new comming feature. it leads to class collision,
- it is very difficult to manage.
- more permutation and combination.



- Decorator has both Has-a and Is-a relationship

Examples:

```
package System_Design.DesignPatterns.Decorator_Design_Pattern.Pizza_Example;

public abstract class BasePizza {

    public abstract int cost();
}
```

```
package System_Design.DesignPatterns.Decorator_Design_Pattern.Pizza_Example;

public class FarmHouse extends BasePizza{
    @Override
    public int cost() {
        return 200;
    }
}
```

```
package System_Design.DesignPatterns.Decorator_Design_Pattern.Pizza_Example;

public class Margherita extends BasePizza{
    @Override
    public int cost() {
        return 100;
    }
}
```

```
package System_Design.DesignPatterns.Decorator_Design_Pattern.Pizza_Example;

public class VegDelight extends BasePizza{
    @Override
    public int cost() {
        return 120;
    }
}
```

```
package System_Design.DesignPatterns.Decorator_Design_Pattern.Pizza_Example;

public abstract class ToppingDecorator extends BasePizza{
}
```

```java
package System_Design.DesignPatterns.Decorator_Design_Pattern.Pizza_Example;

public class MushRoom extends ToppingDecorator{

    BasePizza basePizza;
    public MushRoom(BasePizza pizza){
        this.basePizza=pizza;
    }

    @Override
    public int cost() {
        return this.basePizza.cost()+15;
    }
}
```

```java
package System_Design.DesignPatterns.Decorator_Design_Pattern.Pizza_Example;

public class ExtraCheese extends ToppingDecorator{
    BasePizza basePizza;
    public ExtraCheese(BasePizza pizza){
        this.basePizza=pizza;
    }

    @Override
    public int cost() {
        return this.basePizza.cost()+150;
    }
}
```

```java
package System_Design.DesignPatterns.Decorator_Design_Pattern.Pizza_Example;

public class Main {
    public static void main(String[] args){
        BasePizza pizza = new ExtraCheese(new Margherita());
        System.out.println( "pizza.cost(): "+pizza.cost());

        BasePizza pizza2 = new MushRoom(new ExtraCheese(new Margherita()));
        System.out.println( "pizza2.cost(): "+pizza2.cost());
    }
}
output:
pizza.cost(): 250
pizza2.cost(): 265
```

```
Process finished with exit code 0
```