デザインパターン「2-Adapter」

Seeeeee:D　夏休み勉強会

いつ使うの??

- 関連性のないクラス通しを既存のクラスを変えずに関連付けする場合
- 既存のクラスをある新しいインターフェイスを通じて再利用したい場合
- 既存のクラス・インターフェイスをアプリケーション固有なものに(使いやすく)したい場合

用語説明


Target
Adapter
Adaptee
実際に利用される
インターフェイス
implememts
TargetからAdapteeを使
えるように変換をする
継承
または
委譲
利用される
既存のクラス

Adaptee

もともと提供されるクラス

Rubyのソースコード

```ruby
class Human
  def initialize(name, age)
    @name = name
    @age = age
  end

  def printName
    puts @name
  end

  def printAge
    puts @age
  end
end
```

Javaのソースコード

```java
public class Human {
   private String name;
   private int age;

   public Human(String name, int age) {
      this.name = name;
      this.age = age;
   }

   public void printName() {
      System.out.println(name);
   }

   public void printAge() {
      System.out.println(age);
   }
}
```

Target

実際に利用されるインターフェイス

Rubyのソースコード

```ruby
class Student
  def initialize(adapter)
    @adapter = adapter
  end

  def showName
    @adapter.showName
  end

  def showAge
    @adapter.showAge
  end
end
```

Javaのソースコード

```java
public interface Student {
   public abstract void showName();
   public abstract void showAge();
}
```

必要なインターフェイスを宣言するのみでAdapter関連の実際に処理を行うコードは書かない。

Adapter

変換を行うクラス

Rubyのソースコード

```ruby
require './adaptee'
class HumanAdapter
  def initialize(name,age)
    @human=Human.new(name,age)
  end
  def showName
    @human.printName
  end

  def showAge
    @human.printAge
  end
end
```

Javaのソースコード

```java
public class HumanAdapter extends Human implements Student {
   public HumanAdapter(String name, int age) {
      super(name, age);
   }

    public void showName() {
      printName();
    }

    public void showAge() {
      printAge();
    }
 }
```

必要に応じて変形させながらAdapteeのメゾットを呼び出す感じ

Main

もともと提供されるクラス

Rubyのソースコード

```ruby
require './adapter'
require './target'

student=Student.new(HumanAdapter.new(
"Hotsukai",20
))
student.showName
student.showAge
```

Javaのソースコード

```java
public class Main {
  public static void main(String[] args) {
    Student student = new HumanAdapter(“Hotsukai", 20);
    student.showName();
    student.showAge();
  }
}
```

まとめ

- **既存のクラスを再利用するので車輪の再発明を防げる!**
- **既存のクラスには手を付けないのでバグの原因を絞れる。**
- **既存のクラスには手を付けないので新たな不具合が発生しない。**

- **疑問**

Targetで直接Adapteeのインスタンス作ったり継承するんじゃだめなん??