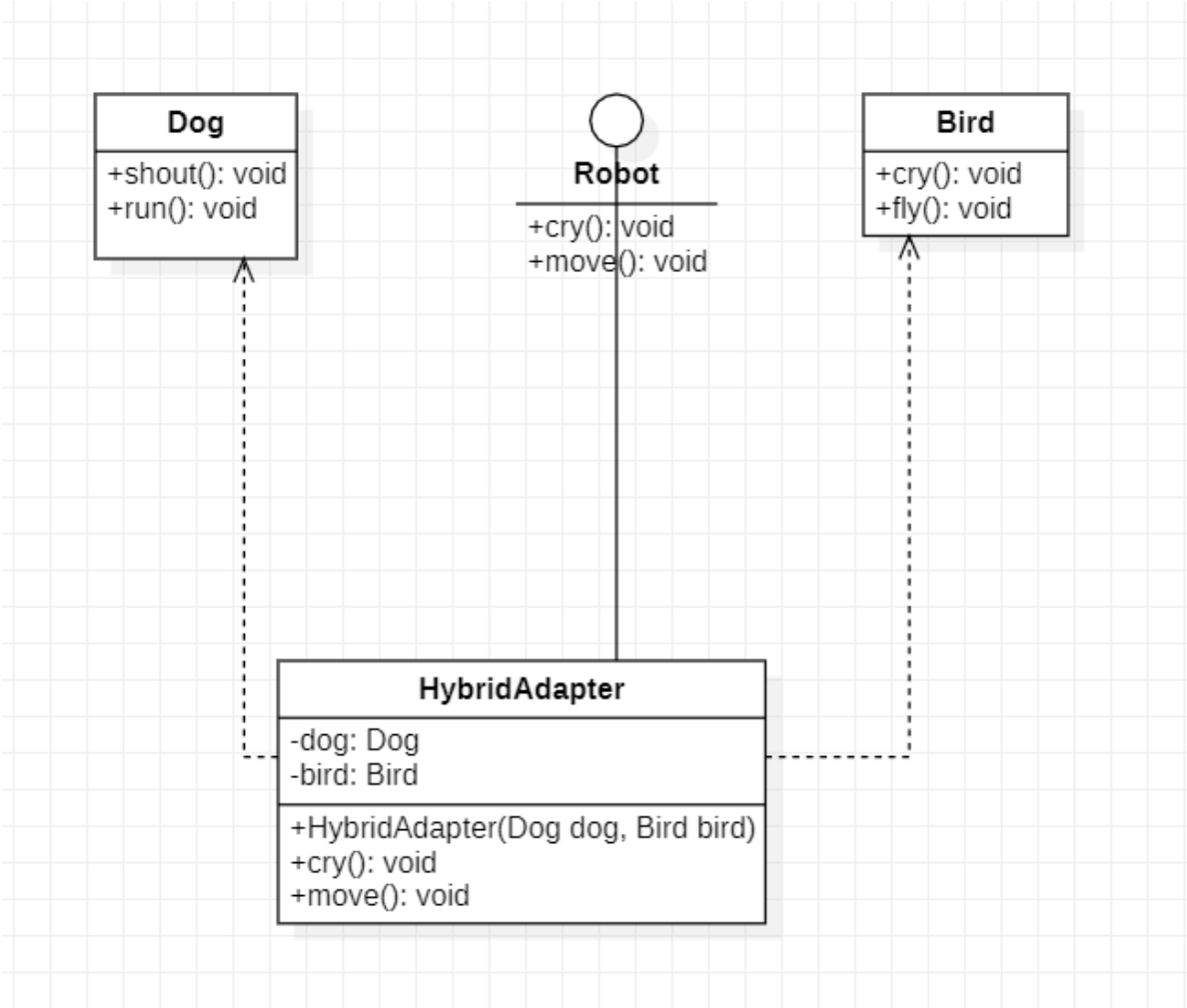
09020328-王亮-设计模式1-6周

Assignment1-Adapter_Object

对象方式的适配器模式

UML



Code

```
package seu.assignment.adapter_object;

/**
 * @ClassName: Bird
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 16:57:28
 * @Input:
 * @Output:
 */
class Bird {
   public void fly() {
      System.out.println("[Bird] Fly like a bird");
```

```
    }
    public void cry() {
        System.out.println("[Bird] Cry like a bird");
    }
}
```

```
package seu.assignment.adapter_object;

/**
 * @ClassName: Dog
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 16:57:22
 * @Input:
 * @Output:
 */
class Dog {
    public void run() {
        System.out.println("[Dog] Run like a dog");
    }
    public void shout() {
        System.out.println("[Dog] Shout like a dog");
    }
}
```

```
package seu.assignment.adapter_object;

public interface Robot {
    void cry();
    void move();
}
```

```
package seu.assignment.adapter_object;

/**
 * @ClassName: HybridAdapter
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 16:57:37
 * @Input:
 * @Output:
 */
class HybridAdapter implements Robot {
    Dog dog;
    Bird bird;
    public HybridAdapter(Dog dog, Bird bird) {
        this.dog = dog;
        this.bird = bird;
    }

    @Override
    public void cry() {
        bird.cry();
```

```
    }

    @Override
    public void move() {
        dog.run();
    }
}
```

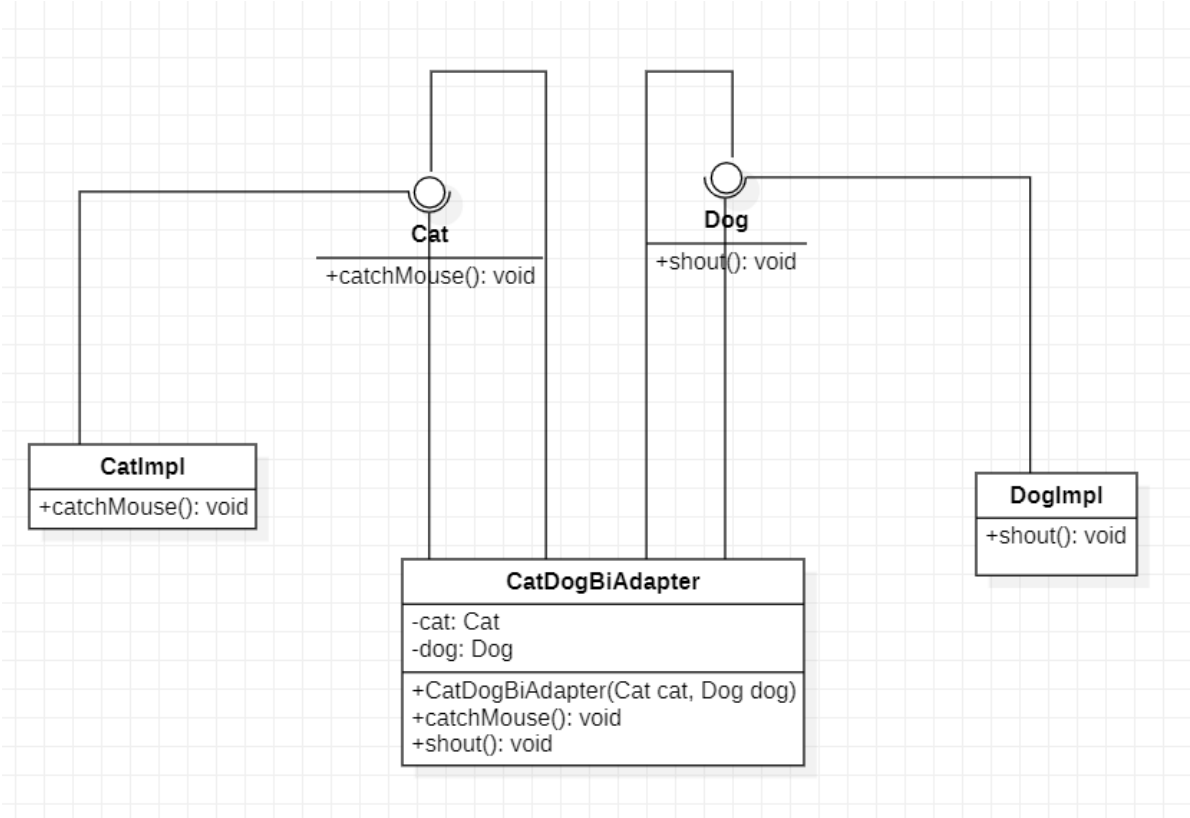
```
package seu.assignment.adapter_object;

/**
 * @ClassName: Client
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 17:01:42
 * @Input:
 * @Output:
 */
class Client {
    public static void main(String[] args) {
        Dog dog = new Dog();
        Bird bird = new Bird();
        Robot robot = new HybridAdapter(dog, bird);
        robot.cry(); // like a bird
        robot.move(); // like a dog
    }
}
```

```
[Bird] Cry like a bird
[Dog] Run like a dog
```

Assignment2-Bi_Adapter

双向适配器

UML



Code

```java
package seu.assignment.bi_adapter;

public interface Cat {
    void catchMouse();
}
```

```java
package seu.assignment.bi_adapter;

public interface Dog {
    void shout();
}
```

```java
package seu.assignment.bi_adapter;

/**
 * @ClassName: CatImpl
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 20:03:48
 * @Input:
 * @Output:
 */
class CatImpl implements Cat {
    @Override
    public void catchMouse() {
        System.out.println("[CatImpl] Catch mouse like a cat");
    }
}
```

```java
package seu.assignment.bi_adapter;

/**
 * @ClassName: DogImpl
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 20:03:55
 * @Input:
 * @Output:
 */
class DogImpl implements Dog {
    @Override
    public void shout() {
        System.out.println("[DogImpl] Shout like a dog");
    }
}
```

```java
package seu.assignment.bi_adapter;

/**
 * @ClassName: CatDogBiAdapter
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 20:16:42
 * @Input:
 * @Output:
 */
class CatDogBiAdapter implements Cat, Dog {
    Cat cat;
    Dog dog;
    public CatDogBiAdapter(Cat cat, Dog dog) {
        this.cat = cat;
        this.dog = dog;
    }
    @Override
    public void catchMouse() {
        dog.shout();
    }

    @Override
    public void shout() {
        cat.catchMouse();
    }
}
```

```java
package seu.assignment.bi_adapter;

/**
 * @ClassName: Client
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 20:22:03
 * @Input:
 * @Output:
```

```
 */
class Client {
    public static void main(String[] args) {
        Cat cat = new CatImpl();
        Dog dog = new DogImpl();
        CatDogBiAdapter biAdapter = new CatDogBiAdapter(cat, dog);
        biAdapter.catchMouse(); // like a dog
        biAdapter.shout(); // like a cat
    }
}
```

```
[DogImpl] Shout like a dog
[CatImpl] Catch mouse like a cat
```

Assignment3-Facade

Client视角：

Goal: getCoffee().drink()

Home: fetchBean() -> boilWater() -> chooseMethod() -> getCoffee().drink()

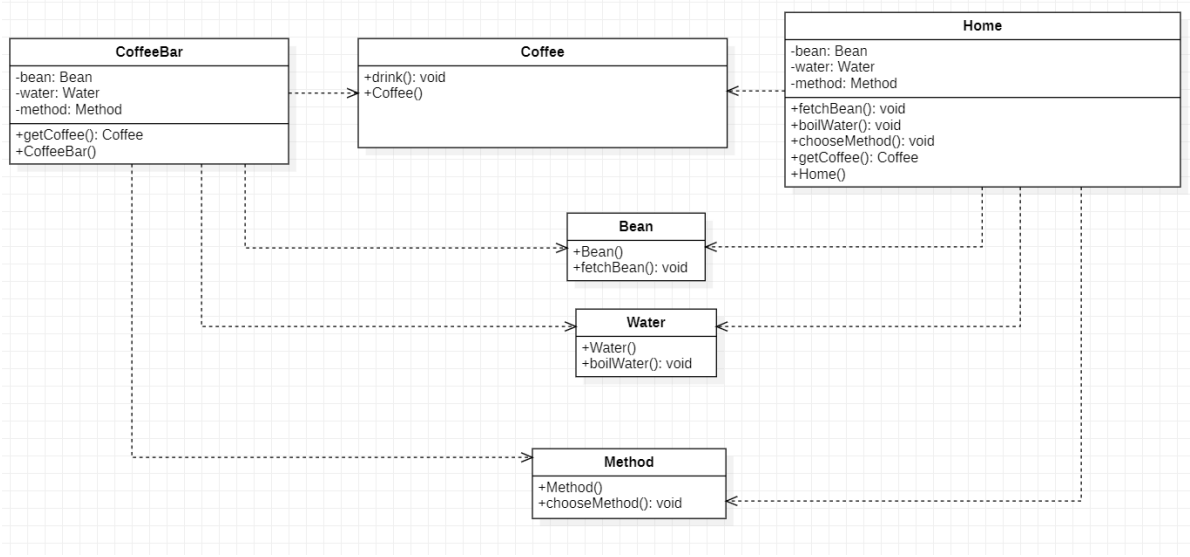
CoffeeBar: getCoffee().drink()

自己泡咖啡需要自行完成所有步骤，才可得到咖啡（自行完成一系列方法调用，才能实现目的）。

去咖啡店直接点咖啡就可以得到咖啡（一系列方法调用封装于接口中，由相应类调用，对外直接调用暴露接口即可实现目的）。

特点上（所有UML都没画Client），对于CoffeeBar，Client只需要依赖这一个类（当然需要使用的Coffee类不计，只计算达到目的所需的类），对于Home，原则上需要Bean Water Method三者类并单独调用方法。

UML



Code

```
package seu.assignment.facade;

/**
 * @ClassName: Bean
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 21:25:26
 * @Input:
 * @Output:
 */
class Bean {
   public Bean() {}
   public void fetchBean() {}
}
```

```
package seu.assignment.facade;

/**
 * @ClassName: Water
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 21:26:25
 * @Input:
 * @Output:
 */
class Water {
   public Water() {}
   public void boilWater() {}
}
```

```
package seu.assignment.facade;

/**
 * @ClassName: Method
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 21:27:49
 * @Input:
 * @Output:
 */
class Method {
   public Method() {}
   public void chooseMethod() {}
}
```

```
package seu.assignment.facade;

/**
 * @ClassName: Coffee
 * @Description: java类描述
 * @Author: 11609
```

```
 * @Date: 2022/9/24 21:28:39
 * @Input:
 * @Output:
 */
class Coffee {
    public Coffee() {}
    public void drink() {
        System.out.println("[Coffee] Exceptional Coffee!");
    }
}
```

```
package seu.assignment.facade;

/**
 * @ClassName: CoffeeBar
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 21:35:37
 * @Input:
 * @Output:
 */
class CoffeeBar {
    public CoffeeBar() {}
    private Bean bean = new Bean();
    private Water water = new Water();
    private Method method = new Method();

    public Coffee getCoffee() {
        bean.fetchBean();
        water.boilWater();
        method.chooseMethod();
        return new Coffee();
    }
}
```

```
package seu.assignment.facade;

/**
 * @ClassName: Home
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 21:32:29
 * @Input:
 * @Output:
 */
class Home {
    private Bean bean = new Bean();
    private Water water = new Water();
    private Method method = new Method();
    public Home() {}
    public void fetchBean() {
        bean.fetchBean();
    }
    public void boilWater() {
```

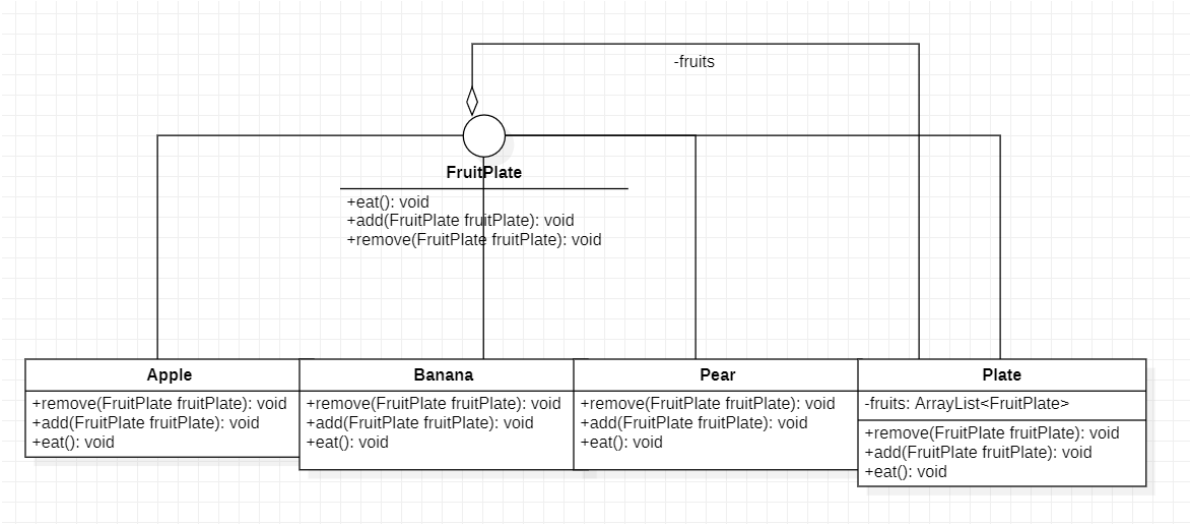
```
        water.boilWater();
    }
    public void chooseMethod() {
        method.chooseMethod();
    }
    public Coffee getCoffee() {
        return new Coffee();
    }
}
```

```
package seu.assignment.facade;

/**
 * @ClassName: Client
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/24 21:48:18
 * @Input:
 * @Output:
 */
class Client {
    public static void main(String[] args) {
        // at home

        Home home = new Home(); // go home
        home.fetchBean();
        home.boilWater();
        home.chooseMethod();
        Coffee coffeeFromHome = home.getCoffee(); // nasty
        coffeeFromHome.drink();

        // at coffee-bar

        CoffeeBar coffeeBar = new CoffeeBar(); // go to coffee bar
        Coffee coffeeFromBar = coffeeBar.getCoffee(); // convenient
        coffeeFromBar.drink();
    }
}
```

```
[Coffee] Exceptional Coffee!
[Coffee] Exceptional Coffee!
```

Assignment4-Composition_Uniformity

UML



将四个类做一并处理，Apple Banana Pear等对eat以外方法作空实现。

Code

```
package seu.assignment.composition_uniformity;

/**
 * @ClassName: Apple
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/25 10:53:05
 * @Input:
 * @Output:
 */
class Apple implements FruitPlate {
    @Override
    public void eat() {
        System.out.println("[Apple] Sweet Apple");
    }

    @Override
    public void add(FruitPlate fruitPlate) {

    }

    @Override
    public void remove(FruitPlate fruitPlate) {

    }
}
```

```
package seu.assignment.composition_uniformity;

/**
 * @ClassName: Banana
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/25 10:53:12
 * @Input:
 * @Output:
```

```
 */
class Banana implements FruitPlate {
    @Override
    public void eat() {
        System.out.println("[Banana] Tasty Banana");
    }

    @Override
    public void add(FruitPlate fruitPlate) {

    }

    @Override
    public void remove(FruitPlate fruitPlate) {

    }
}
```

```
package seu.assignment.composition_uniformity;

/**
 * @ClassName: Pear
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/25 10:53:18
 * @Input:
 * @Output:
 */
class Pear implements FruitPlate {
    @Override
    public void eat() {
        System.out.println("[Pear] Fragrant Pear");
    }

    @Override
    public void add(FruitPlate fruitPlate) {

    }

    @Override
    public void remove(FruitPlate fruitPlate) {

    }
}
```

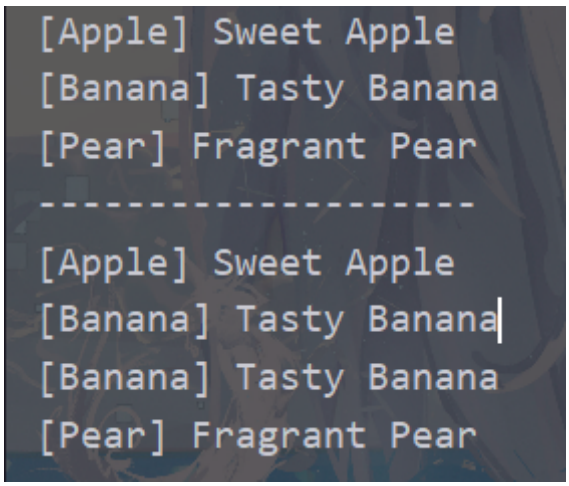
```
package seu.assignment.composition_uniformity;

import java.util.ArrayList;

/**
 * @ClassName: Plate
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/25 10:53:42
```

```
 * @Input:
 * @Output:
 */
class Plate implements FruitPlate {
    private ArrayList<FruitPlate> fruits = new ArrayList<>();
    @Override
    public void eat() {
        fruits.forEach(FruitPlate::eat);
    }

    @Override
    public void add(FruitPlate fruitPlate) {
        fruits.add(fruitPlate);
    }

    @Override
    public void remove(FruitPlate fruitPlate) {
        fruits.remove(fruitPlate);
    }
}
```

```
package seu.assignment.composition_uniformity;

public interface FruitPlate {
   void eat();
   void add(FruitPlate fruitPlate);
   void remove(FruitPlate fruitPlate);
}
```

```
package seu.assignment.composition_uniformity;

/**
 * @ClassName: Client
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/9/25 11:03:59
 * @Input:
 * @Output:
 */
class Client {
   public static void main(String[] args) {
      // prepare some fruits
      FruitPlate apple = new Apple();
      FruitPlate banana1 = new Banana();
      FruitPlate banana2 = new Banana();
      FruitPlate pear = new Pear();

      // add to the plate
      FruitPlate fruitPlate = new Plate();
      fruitPlate.add(apple);
      fruitPlate.add(banana1);
      fruitPlate.add(banana2);
      fruitPlate.add(pear);
```

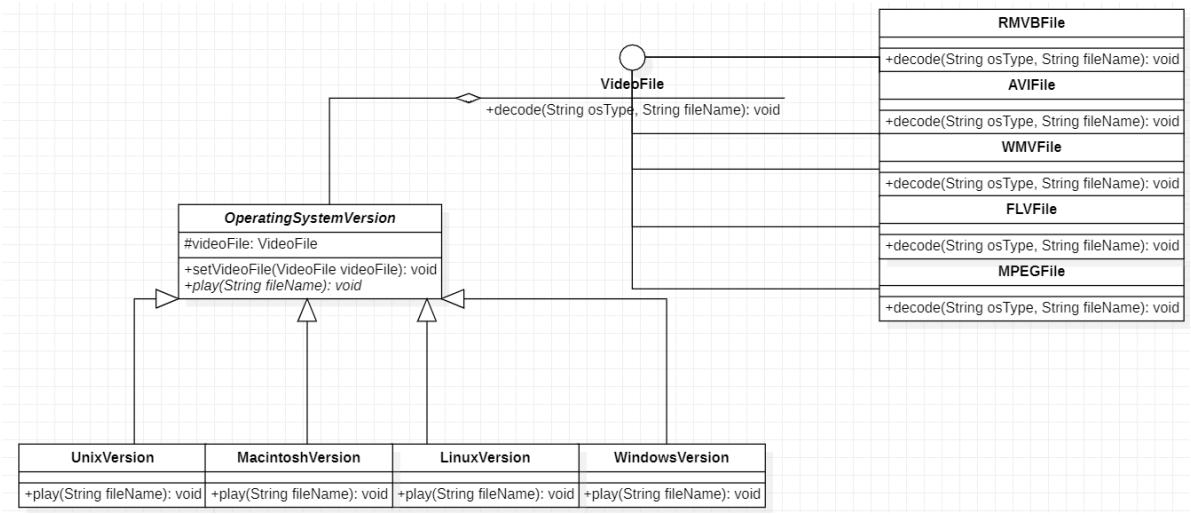
```
        // remove duplicate banana and eat the rest
        fruitPlate.remove(banana2);
        fruitPlate.eat(); // apple -> banana -> pear

        System.out.println("--------------------");

        // eat respectively
        apple.eat();
        banana1.eat();
        banana2.eat();
        pear.eat();
    }
}
```



Assignment5-Bridge

排列组合各多一项

UML



CODE

```
package seu.assignment.bridge;

/**
 * @ClassName: AVIFile
```

```
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:06:21
 * @Input:
 * @Output:
 */
class AVIFile implements VideoFile {
   @Override
   public void decode(String osType, String fileName) {
      System.out.println("OS-Type: " + osType + "------- " + "FileName: " +
fileName + ".avi");
   }
}
```

```
package seu.assignment.bridge;

/**
 * @ClassName: FLVFile
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:06:41
 * @Input:
 * @Output:
 */
class FLVFile implements VideoFile {
   @Override
   public void decode(String osType, String fileName) {
      System.out.println("OS-Type: " + osType + "------- " + "FileName: " +
fileName + ".flv");

   }
}
```

```
package seu.assignment.bridge;

/**
 * @ClassName: MPEGFile
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:07:02
 * @Input:
 * @Output:
 */
class MPEGFile implements VideoFile {
   @Override
   public void decode(String osType, String fileName) {
      System.out.println("OS-Type: " + osType + "------- " + "FileName: " +
fileName + ".mpeg");

   }
}
```

```java
package seu.assignment.bridge;

/**
 * @ClassName: RMVBFile
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:06:15
 * @Input:
 * @Output:
 */
class RMVBFile implements VideoFile {
   @Override
   public void decode(String osType, String fileName) {
      System.out.println("OS-Type: " + osType + "------- " + "FileName: " +
fileName + ".rmvb");

   }
}
```

```java
package seu.assignment.bridge;

/**
 * @ClassName: WMVFile
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:06:28
 * @Input:
 * @Output:
 */
class WMVFile implements VideoFile {
   @Override
   public void decode(String osType, String fileName) {
      System.out.println("OS-Type: " + osType + "------- " + "FileName: " +
fileName + ".wmv");
   }
}
```

```java
package seu.assignment.bridge;

public interface VideoFile {
   void decode(String osType, String fileName);
}
```

```java
package seu.assignment.bridge;

/**
 * @ClassName: LinuxVersion
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:04:40
 * @Input:
 * @Output:
```

```
 */
class LinuxVersion extends OperatingSystemVersion {
   @Override
   public void play(String filename) {
      this.videoFile.decode("Linux", filename);
   }
}
```

```
package seu.assignment.bridge;

/**
 * @ClassName: MacintoshVersion
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:04:33
 * @Input:
 * @Output:
 */
class MacintoshVersion extends OperatingSystemVersion {
   @Override
   public void play(String filename) {
      this.videoFile.decode("Macintosh", filename);

   }
}
```

```
package seu.assignment.bridge;

/**
 * @ClassName: WindowsVersion
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:04:47
 * @Input:
 * @Output:
 */
class WindowsVersion extends OperatingSystemVersion {
   @Override
   public void play(String filename) {
      this.videoFile.decode("Windows", filename);

   }
}
```

```
package seu.assignment.bridge;

/**
 * @ClassName: UnixVersion
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:04:23
 * @Input:
 * @Output:
```

```
 */
class UnixVersion extends OperatingSystemVersion {
   @Override
   public void play(String filename) {
      this.videoFile.decode("Unix", filename);

   }
}
```

```
package seu.assignment.bridge;

abstract class OperatingSystemVersion {
   protected VideoFile videoFile;

   public void setVideoFile(VideoFile videoFile) {
      this.videoFile = videoFile;
   }

   public abstract void play(String filename);
}
```

```
package seu.assignment.bridge;

import java.util.ArrayList;
import java.util.List;

/**
 * @ClassName: Client
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:12:27
 * @Input:
 * @Output:
 */
class Client {
   public static void main(String[] args) {
      List<VideoFile> videoList = new ArrayList<>();
      VideoFile aviFile = new AVIFile();
      VideoFile flvFile = new FLVFile();
      VideoFile mpegFile = new MPEGFile();
      VideoFile rmvbFile = new RMVBFile();
      VideoFile wmvFile = new WMVFile();

      videoList.add(aviFile);
      videoList.add(flvFile);
      videoList.add(mpegFile);
      videoList.add(rmvbFile);
      videoList.add(wmvFile);

      List<OperatingSystemVersion> osList = new ArrayList<>();
      OperatingSystemVersion linuxVersion = new LinuxVersion();
      OperatingSystemVersion macintoshVersion = new MacintoshVersion();
      OperatingSystemVersion windowsVersion = new WindowsVersion();
      OperatingSystemVersion unixVersion = new UnixVersion();
```

```
        osList.add(linuxVersion);
        osList.add(macintoshVersion);
        osList.add(windowsVersion);
        osList.add(unixVersion);

        for (OperatingSystemVersion os : osList) {
            for (VideoFile video : videoList) {
                os.setVideoFile(video);
                os.play("Camelia");
            }
        }

    }
}
```

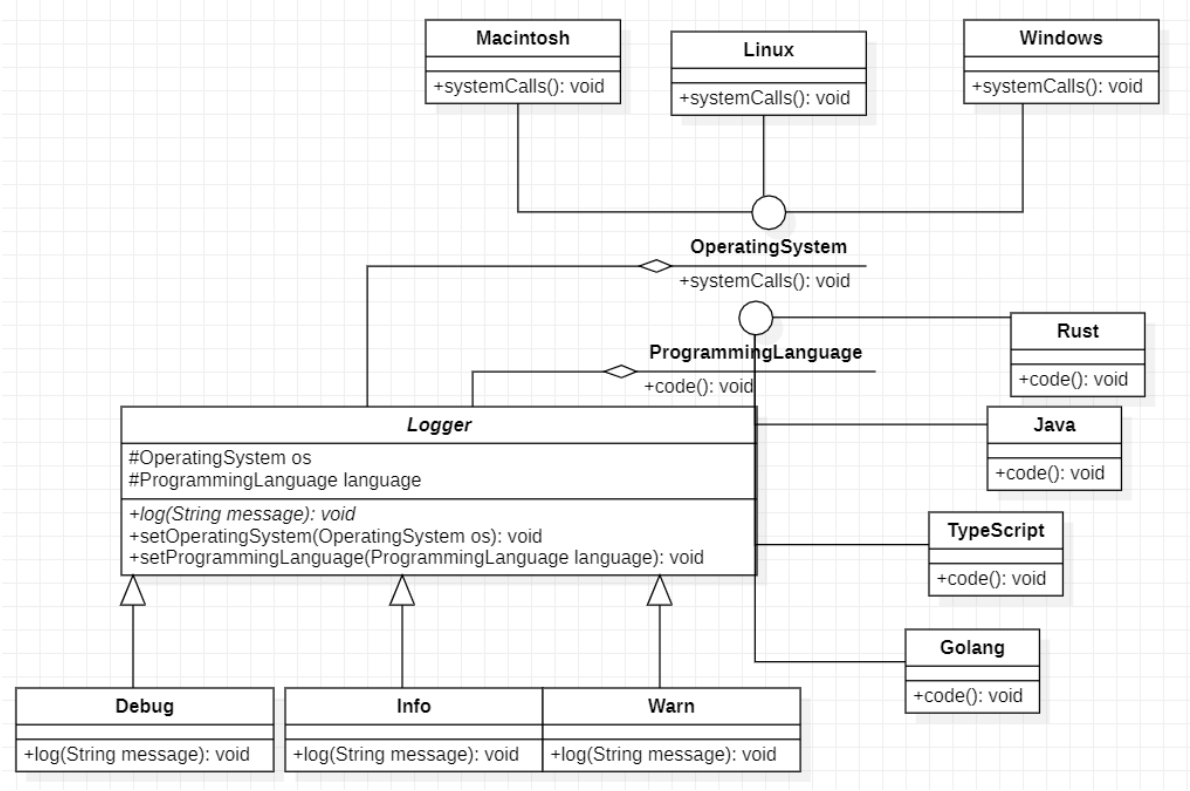
```
OS-Type: Linux------- FileName: Camelia.avi
OS-Type: Linux------- FileName: Camelia.flv
OS-Type: Linux------- FileName: Camelia.mpeg
OS-Type: Linux------- FileName: Camelia.rmvb
OS-Type: Linux------- FileName: Camelia.wmv
OS-Type: Macintosh------- FileName: Camelia.avi
OS-Type: Macintosh------- FileName: Camelia.flv
OS-Type: Macintosh------- FileName: Camelia.mpeg
OS-Type: Macintosh------- FileName: Camelia.rmvb
OS-Type: Macintosh------- FileName: Camelia.wmv
```

```
OS-Type: Windows------- FileName: Camelia.avi
OS-Type: Windows------- FileName: Camelia.flv
OS-Type: Windows------- FileName: Camelia.mpeg
OS-Type: Windows------- FileName: Camelia.rmvb
OS-Type: Windows------- FileName: Camelia.wmv
OS-Type: Unix------- FileName: Camelia.avi
OS-Type: Unix------- FileName: Camelia.flv
OS-Type: Unix------- FileName: Camelia.mpeg
OS-Type: Unix------- FileName: Camelia.rmvb
OS-Type: Unix------- FileName: Camelia.wmv
```

Assignment6-Bridge_Triple

三个空位的排列组合

UML



Code

```java
package seu.assignment.triple_bridge;

/**
 * @ClassName: Windows
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:59:40
 * @Input:
 * @Output:
 */
class Windows implements OperatingSystem {
   @Override
   public void systemCalls() {
      System.out.println("Runs on: -> Windows");

   }
}
```

```java
package seu.assignment.triple_bridge;

/**
 * @ClassName: Linux
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:59:15
 * @Input:
 * @Output:
 */
```

```
class Linux implements OperatingSystem {
   @Override
   public void systemCalls() {
      System.out.println("Runs on: -> Linux");

   }
}
```

```
package seu.assignment.triple_bridge;

/**
 * @ClassName: Macintosh
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:59:01
 * @Input:
 * @Output:
 */
class Macintosh implements OperatingSystem {
   @Override
   public void systemCalls() {
      System.out.println("Runs on: -> Macintosh");

   }
}
```

```
package seu.assignment.triple_bridge;

public interface OperatingSystem {
   void systemCalls();
}
```

```
package seu.assignment.triple_bridge;

/**
 * @ClassName: Java
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 21:00:07
 * @Input:
 * @Output:
 */
class Java implements ProgrammingLanguage {
   @Override
   public void code() {
      System.out.println("Programmed in: -> Java");
   }
}
```

```
package seu.assignment.triple_bridge;

/**
 * @ClassName: TypeScript
```

```
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 21:00:22
 * @Input:
 * @Output:
 */
class TypeScript implements ProgrammingLanguage {
    @Override
    public void code() {
        System.out.println("Programmed in: -> TypeScript");

    }
}
```

```
package seu.assignment.triple_bridge;

/**
 * @ClassName: Rust
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:59:55
 * @Input:
 * @Output:
 */
class Rust implements ProgrammingLanguage {
    @Override
    public void code() {
        System.out.println("Programmed in: -> Rust");

    }
}
```

```
package seu.assignment.triple_bridge;

/**
 * @ClassName: Golang
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 21:00:36
 * @Input:
 * @Output:
 */
class Golang implements ProgrammingLanguage {
    @Override
    public void code() {
        System.out.println("Programmed in: -> Golang");
    }
}
```

```java
package seu.assignment.triple_bridge;

public interface ProgrammingLanguage {
   void code();
}
```

```java
package seu.assignment.triple_bridge;

/**
 * @ClassName: Debug
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:56:16
 * @Input:
 * @Output:
 */
class Debug extends Logger {
   @Override
   public void log(String message) {
      System.out.println("Debug: ------------------------");
      System.out.println("Message: -> " + message);
      language.code();
      os.systemCalls();
      System.out.println("-------------------------------");
      System.out.println();
   }
}
```

```java
package seu.assignment.triple_bridge;

/**
 * @ClassName: Info
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:56:32
 * @Input:
 * @Output:
 */
class Info extends Logger {
   @Override
   public void log(String message) {
      System.out.println("Info: ------------------------");
      System.out.println("Message: -> " + message);
      language.code();
      os.systemCalls();
      System.out.println("-------------------------------");
      System.out.println();
   }
}
```

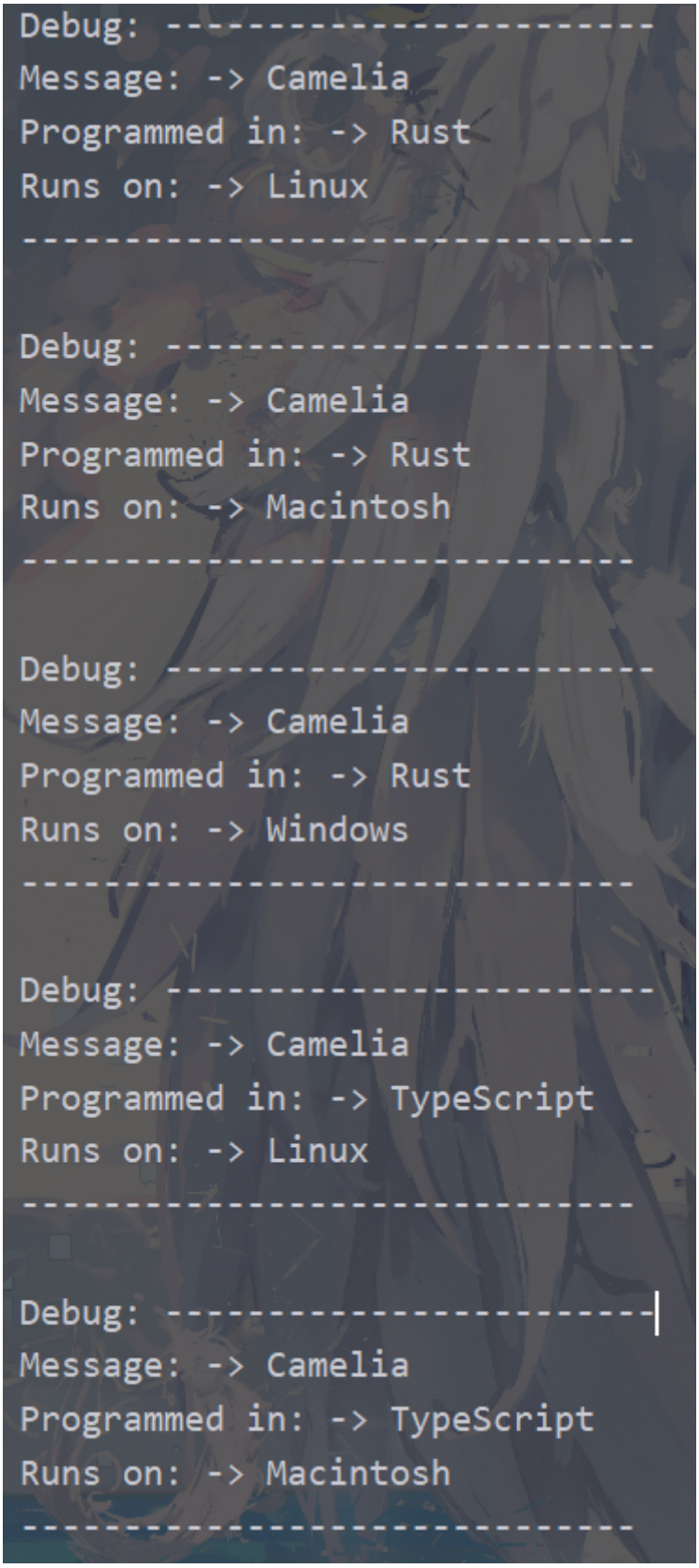
```java
package seu.assignment.triple_bridge;

/**
```

```
 * @ClassName: Warn
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:56:57
 * @Input:
 * @Output:
 */
class Warn extends Logger {
    @Override
    public void log(String message) {
        System.out.println("Warn: ------------------------");
        System.out.println("Message: -> " + message);
        language.code();
        os.systemCalls();
        System.out.println("-------------------------------");
        System.out.println();
    }
}
```

```
package seu.assignment.triple_bridge;

/**
 * @ClassName: Logger
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 20:49:26
 * @Input:
 * @Output:
 */
abstract class Logger {
    protected OperatingSystem os;
    protected ProgrammingLanguage language;

    public abstract void log(String message);
    public void setOperatingSystem(OperatingSystem os) {
        this.os = os;
    }
    public void setProgrammingLanguage(ProgrammingLanguage language) {
        this.language = language;
    }
}
```

```
package seu.assignment.triple_bridge;
import java.util.ArrayList;
import java.util.List;

/**
 * @ClassName: Client
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 21:03:34
 * @Input:
 * @Output:
 */
```

```
class Client {
   public static void main(String[] args) {
      List<Logger> loggerList = new ArrayList<Logger>();
      Logger debug = new Debug();
      Logger warn = new Warn();
      Logger info = new Info();
      loggerList.add(debug);
      loggerList.add(warn);
      loggerList.add(info);

      List<OperatingSystem> osList = new ArrayList<>();
      OperatingSystem linux = new Linux();
      OperatingSystem macintosh = new Macintosh();
      OperatingSystem windows = new Windows();
      osList.add(linux);
      osList.add(macintosh);
      osList.add(windows);

      List<ProgrammingLanguage> languageList = new ArrayList<>();
      ProgrammingLanguage rust = new Rust();
      ProgrammingLanguage typeScript = new TypeScript();
      ProgrammingLanguage java = new Java();
      ProgrammingLanguage golang = new Golang();
      languageList.add(rust);
      languageList.add(typeScript);
      languageList.add(java);
      languageList.add(golang);

      for (Logger logger : loggerList) {
         for (ProgrammingLanguage language : languageList) {
            for (OperatingSystem os : osList) {
               logger.setOperatingSystem(os);
               logger.setProgrammingLanguage(language);
               logger.log("Camelia");
            }
         }
      }
   }
}
```


Debug: ------------------------
Message: -> Camelia
Programmed in: -> Rust
Runs on: -> Linux

Debug: ------------------------
Message: -> Camelia
Programmed in: -> Rust
Runs on: -> Macintosh

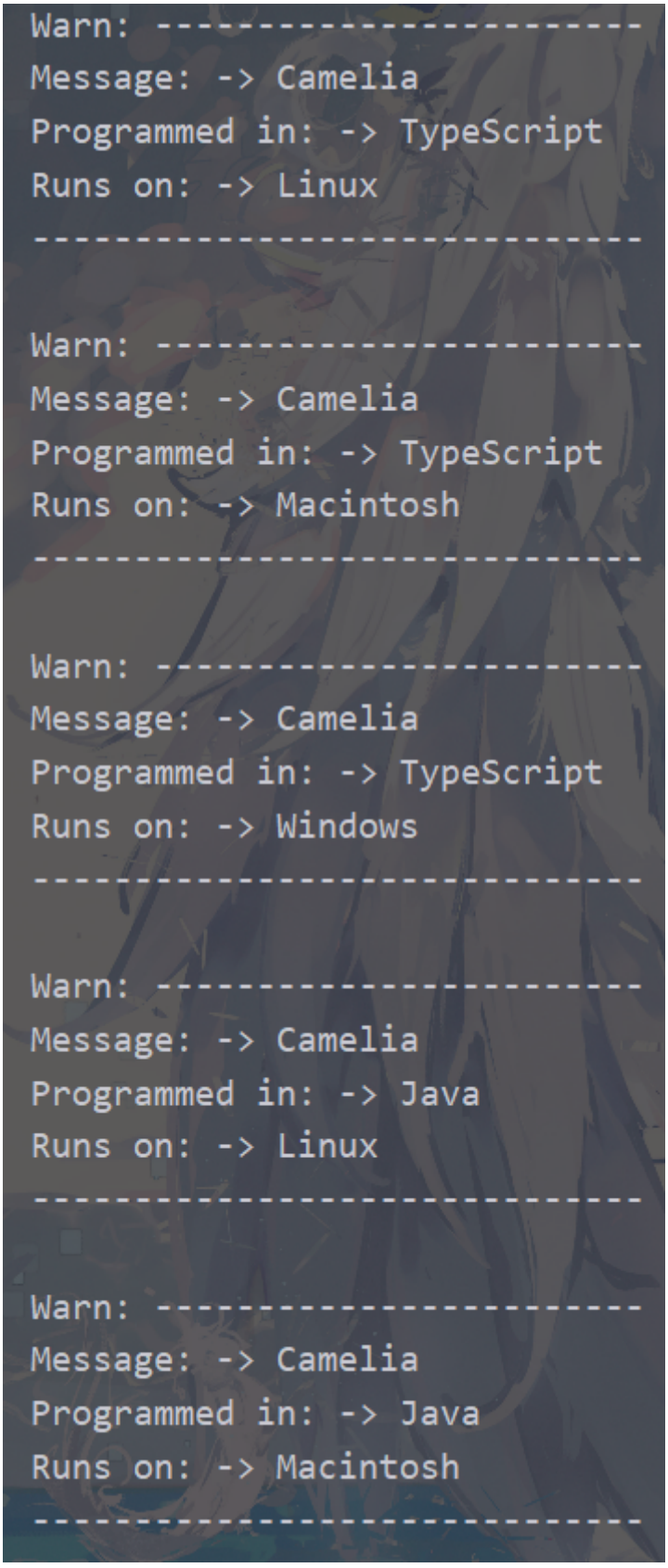
Debug: ------------------------
Message: -> Camelia
Programmed in: -> Rust
Runs on: -> Windows

Debug: ------------------------
Message: -> Camelia
Programmed in: -> TypeScript
Runs on: -> Linux

Debug: ------------------------
Message: -> Camelia
Programmed in: -> TypeScript
Runs on: -> Macintosh

```
Debug: ------------------------
Message: -> Camelia
Programmed in: -> TypeScript
Runs on: -> Windows
------------------------------

Debug: ------------------------
Message: -> Camelia
Programmed in: -> Java
Runs on: -> Linux
------------------------------

Debug: ------------------------
Message: -> Camelia
Programmed in: -> Java
Runs on: -> Macintosh
------------------------------

Debug: ------------------------
Message: -> Camelia
Programmed in: -> Java
Runs on: -> Windows
------------------------------

Debug: ------------------------
Message: -> Camelia
Programmed in: -> Golang
Runs on: -> Linux
------------------------------
```

```
Debug: ------------------------
Message: -> Camelia
Programmed in: -> Golang
Runs on: -> Macintosh
------------------------------

Debug: ------------------------
Message: -> Camelia
Programmed in: -> Golang
Runs on: -> Windows
------------------------------

Warn: -------------------------
Message: -> Camelia
Programmed in: -> Rust
Runs on: -> Linux
------------------------------

Warn: -------------------------
Message: -> Camelia
Programmed in: -> Rust
Runs on: -> Macintosh
------------------------------

Warn: -------------------------
Message: -> Camelia
Programmed in: -> Rust
Runs on: -> Windows
------------------------------
```


Warn: ------------------------
Message: -> Camelia
Programmed in: -> TypeScript
Runs on: -> Linux

Warn: ------------------------
Message: -> Camelia
Programmed in: -> TypeScript
Runs on: -> Macintosh

Warn: ------------------------
Message: -> Camelia
Programmed in: -> TypeScript
Runs on: -> Windows

Warn: ------------------------
Message: -> Camelia
Programmed in: -> Java
Runs on: -> Linux

Warn: ------------------------
Message: -> Camelia
Programmed in: -> Java
Runs on: -> Macintosh

```
Warn: ------------------------
Message: -> Camelia
Programmed in: -> Java
Runs on: -> Windows
------------------------------

Warn: ------------------------
Message: -> Camelia
Programmed in: -> Golang
Runs on: -> Linux
------------------------------

Warn: ------------------------
Message: -> Camelia
Programmed in: -> Golang
Runs on: -> Macintosh
------------------------------

Warn: ------------------------
Message: -> Camelia
Programmed in: -> Golang
Runs on: -> Windows
------------------------------

Info: ------------------------
Message: -> Camelia
Programmed in: -> Rust
Runs on: -> Linux
------------------------------
```

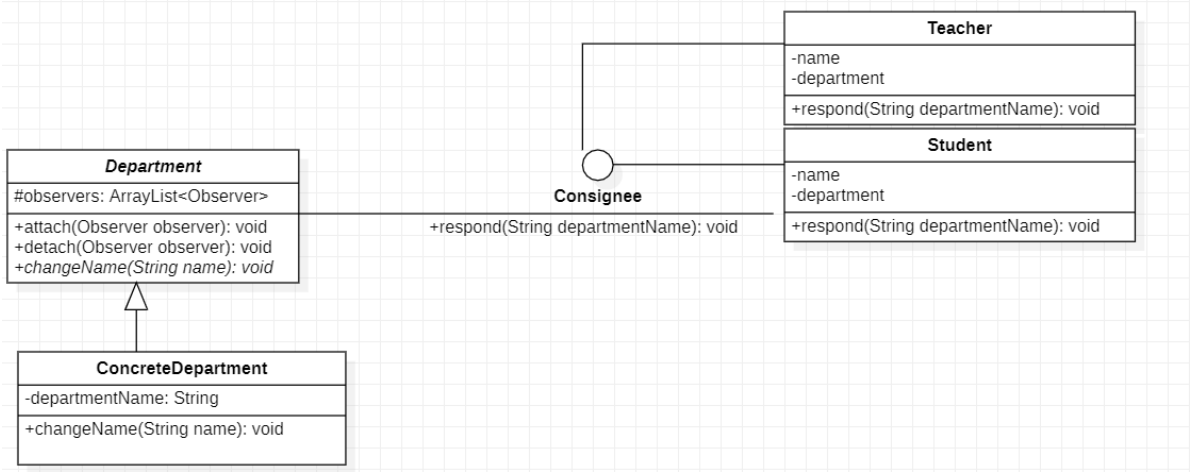
```
Info: ------------------------
Message: -> Camelia
Programmed in: -> Rust
Runs on: -> Macintosh
------------------------------

Info: ------------------------
Message: -> Camelia
Programmed in: -> Rust
Runs on: -> Windows
------------------------------

Info: ------------------------
Message: -> Camelia
Programmed in: -> TypeScript
Runs on: -> Linux
------------------------------

Info: ------------------------
Message: -> Camelia
Programmed in: -> TypeScript
Runs on: -> Macintosh
------------------------------

Info: ------------------------
Message: -> Camelia
Programmed in: -> TypeScript
Runs on: -> Windows
------------------------------
```

```
Info: ------------------------
Message: -> Camelia
Programmed in: -> Java
Runs on: -> Linux
------------------------------

Info: ------------------------
Message: -> Camelia
Programmed in: -> Java
Runs on: -> Macintosh
------------------------------

Info: ------------------------
Message: -> Camelia
Programmed in: -> Java
Runs on: -> Windows
------------------------------

Info: ------------------------
Message: -> Camelia
Programmed in: -> Golang
Runs on: -> Linux
------------------------------

Info: ------------------------
Message: -> Camelia
Programmed in: -> Golang
Runs on: -> Macintosh
------------------------------
```

```
Info: ------------------------
Message: -> Camelia
Programmed in: -> Golang
Runs on: -> Windows
------------------------------
```

Assignment7-Observer

UML



Code

```
package seu.assignment.observer;

/**
 * @ClassName: Teacher
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 21:53:47
 * @Input:
 * @Output:
 */
class Teacher implements Consignee {
   private String name;
   private String department;

   public Teacher() {};
   public Teacher(String name) {
      this.name = name;
   }
   @Override
   public void respond(String departmentName) {
      this.department = departmentName;
      System.out.println("Teacher: " + name + " -> Department: " + department +
" ! ---------");
   }
}
```

```
package seu.assignment.observer;

/**
 * @ClassName: Student
 * @Description: java类描述
 * @Author: 11609
```

```
 * @Date: 2022/10/4 21:53:18
 * @Input:
 * @Output:
 */
class Student implements Consignee {
   private String name;
   private String department;

   public Student(){}
   public Student(String name) {
      this.name = name;
   }
   @Override
   public void respond(String departmentName) {
      this.department = departmentName;
      System.out.println("Student: " + name + " -> Department: " + department +
" ! ---------");
   }
}
```

```
package seu.assignment.observer;

public interface Consignee {
   void respond(String departmentName);
}
```

```
package seu.assignment.observer;

import java.util.ArrayList;
import java.util.List;
import java.util.Observer;

/**
 * @ClassName: Department
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 21:50:14
 * @Input:
 * @Output:
 */
abstract class Department {
   protected List<Consignee> observers = new ArrayList<>();
   public void attach(Consignee observer) {
      this.observers.add(observer);
   }
   public void detach(Consignee observer) {
      this.observers.remove(observer);
   }
   public abstract void changeName(String name);
}
```

```
package seu.assignment.observer;
```

```java
/**
 * @ClassName: ConcreteDepartment
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 22:00:15
 * @Input:
 * @Output:
 */
class ConcreteDepartment extends Department {
    private String departmentName;


    @Override
    public void changeName(String name) {
        if (name.equals(departmentName)) {
            return;
        }
        this.departmentName = name;
        observers.forEach(item -> item.respond(departmentName));
    }
}
```

```java
package seu.assignment.observer;

import java.util.ArrayList;
import java.util.List;

/**
 * @ClassName: Client
 * @Description: java类描述
 * @Author: 11609
 * @Date: 2022/10/4 22:00:29
 * @Input:
 * @Output:
 */
class Client {
    public static void main(String[] args) {
        Department department = new ConcreteDepartment();
        List<Teacher> teacherList = new ArrayList<>();
        Teacher jupyter = new Teacher("Jupyter");
        Teacher camelia = new Teacher("Camelia");
        Teacher ellie = new Teacher("Ellie");
        Teacher roxana = new Teacher("Roxana");
        Teacher vivienne = new Teacher("Vivienne");

        teacherList.add(jupyter);
        teacherList.add(camelia);
        teacherList.add(ellie);
        teacherList.add(roxana);
        teacherList.add(vivienne);

        List<Student> studentList = new ArrayList<>();
        Student ruby = new Student("Ruby");
        Student fiona = new Student("Fiona");
        Student illyana = new Student("Illyana");
```

```
        Student helga = new Student("Helga");

        studentList.add(ruby);
        studentList.add(fiona);
        studentList.add(illyana);
        studentList.add(helga);

        teacherList.forEach(department::attach);
        studentList.forEach(department::attach);
        System.out.println("Switch to -------------- AI");
        department.changeName("Artificial Intelligence");
        System.out.println();
        System.out.println("Switch to -------------- CS");
        department.changeName("Computer Science");
    }
}
```