Extra övningar

Designmönster, analys och design

Utbildare: Mahmud Al Hakim

NACKADEMIN

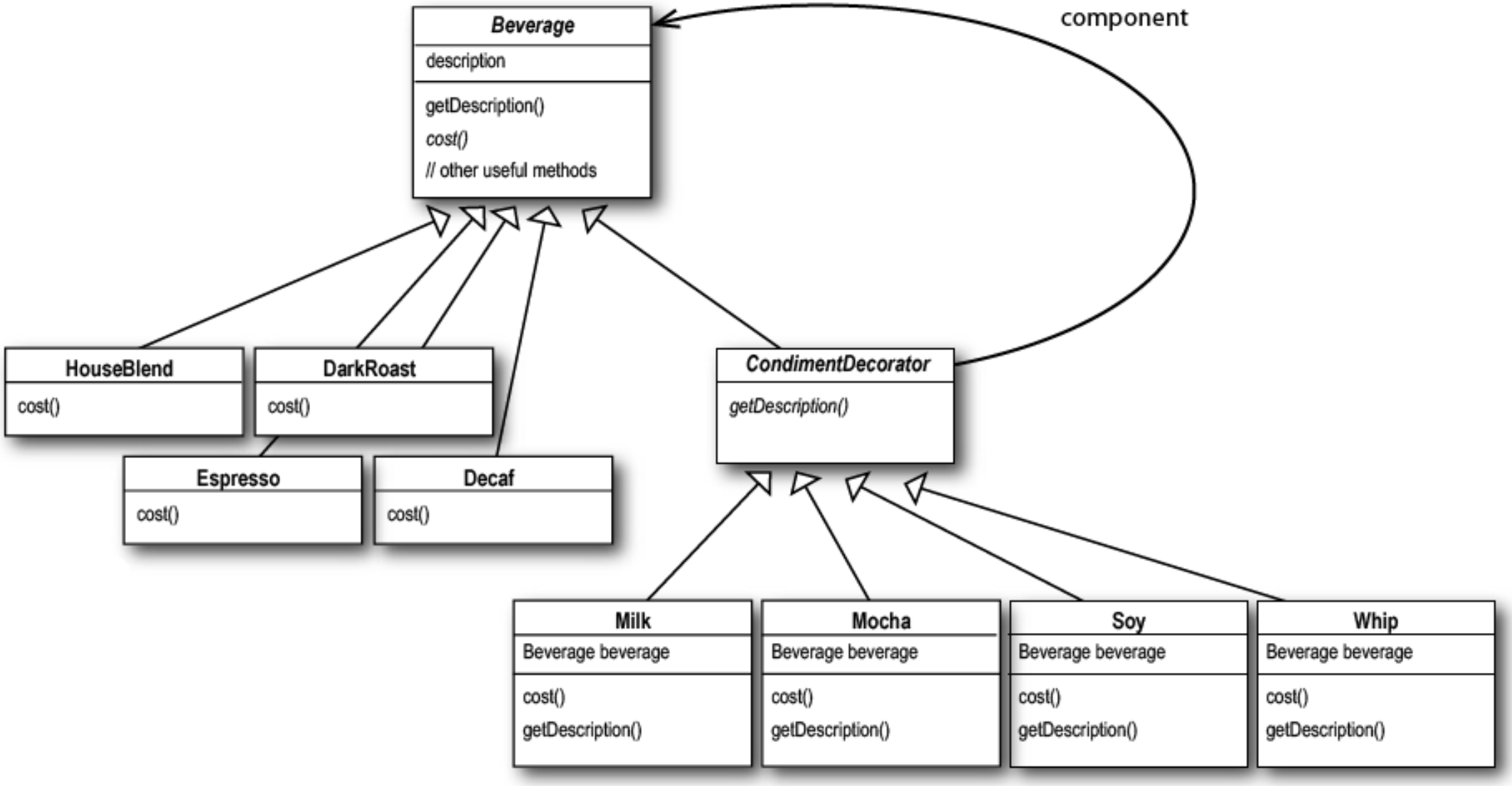
Övning 1

Använder klassen **Calendar** i standardbiblioteket designmönstret Singleton eller Factory? Varför?

```
var date = Calendar.getInstance();
```

https://stackoverflow.com/questions/1673841/examples-of-gof-design-patterns-in-javas-core-libraries/2707195#2707195

Övning 2: Undersök nedanstående klassdiagram. Vad är detta för designmönster? Implementera klassdiagramet (Tips: se nästa sida)

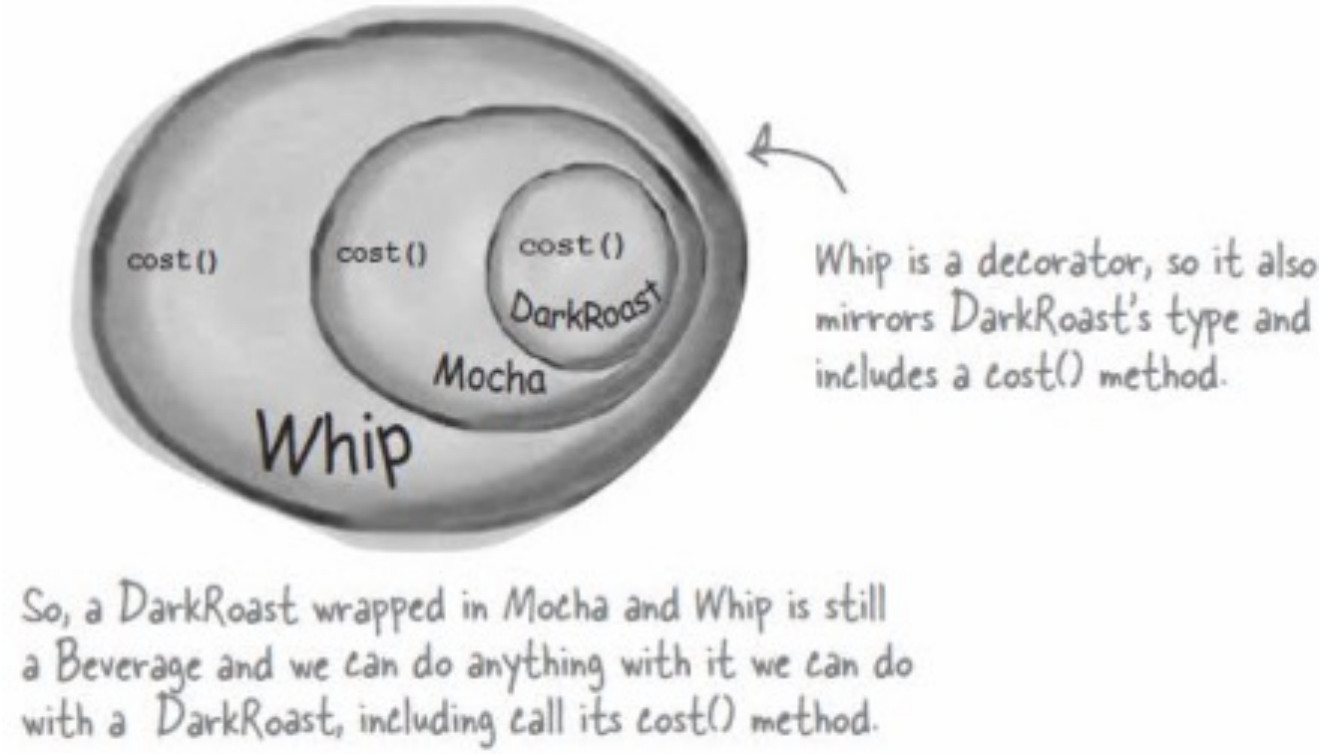


Bildkälla: Head First Design Patterns

Övning 2: exempelkod från main-metoden

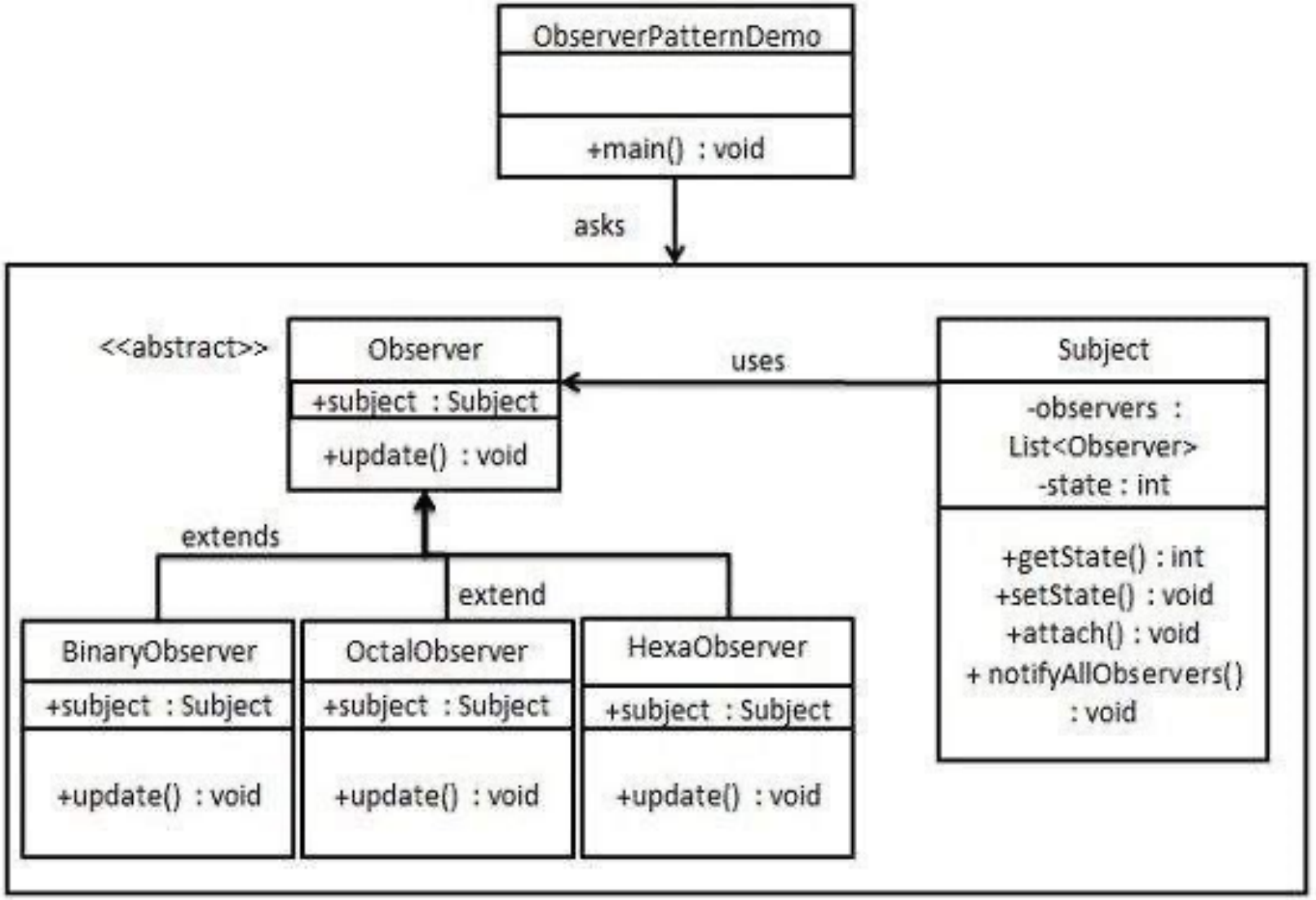
```
Beverage beverage = new Whip(new Mocha(new DarkRoast()));

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

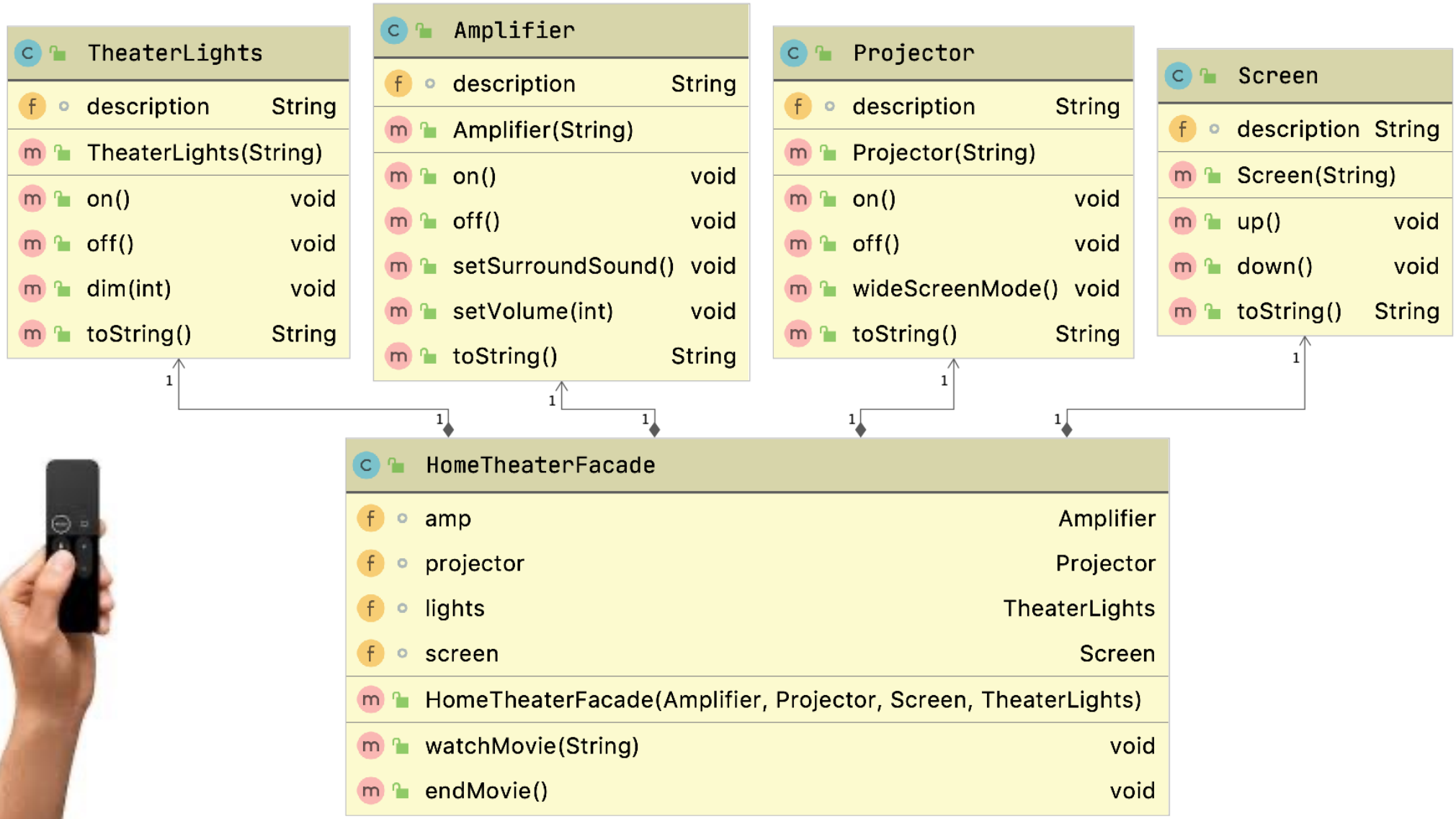


Övning 3: Detta är Observer pattern

Behöver vi verkligen ha en referens till subject i alla observers?
Implementera en ny lösning utan att ha subject i Observer-klassen.
Ändra även den abstrakta klassen **Observer** till ett interface istället.



Övning 4: Undersök nedanstående klassdiagram. Vad är detta för designmönster? Implementera klassdiagramet (Tips: se nästa sida)



Övning 4 – Exempelkod i main

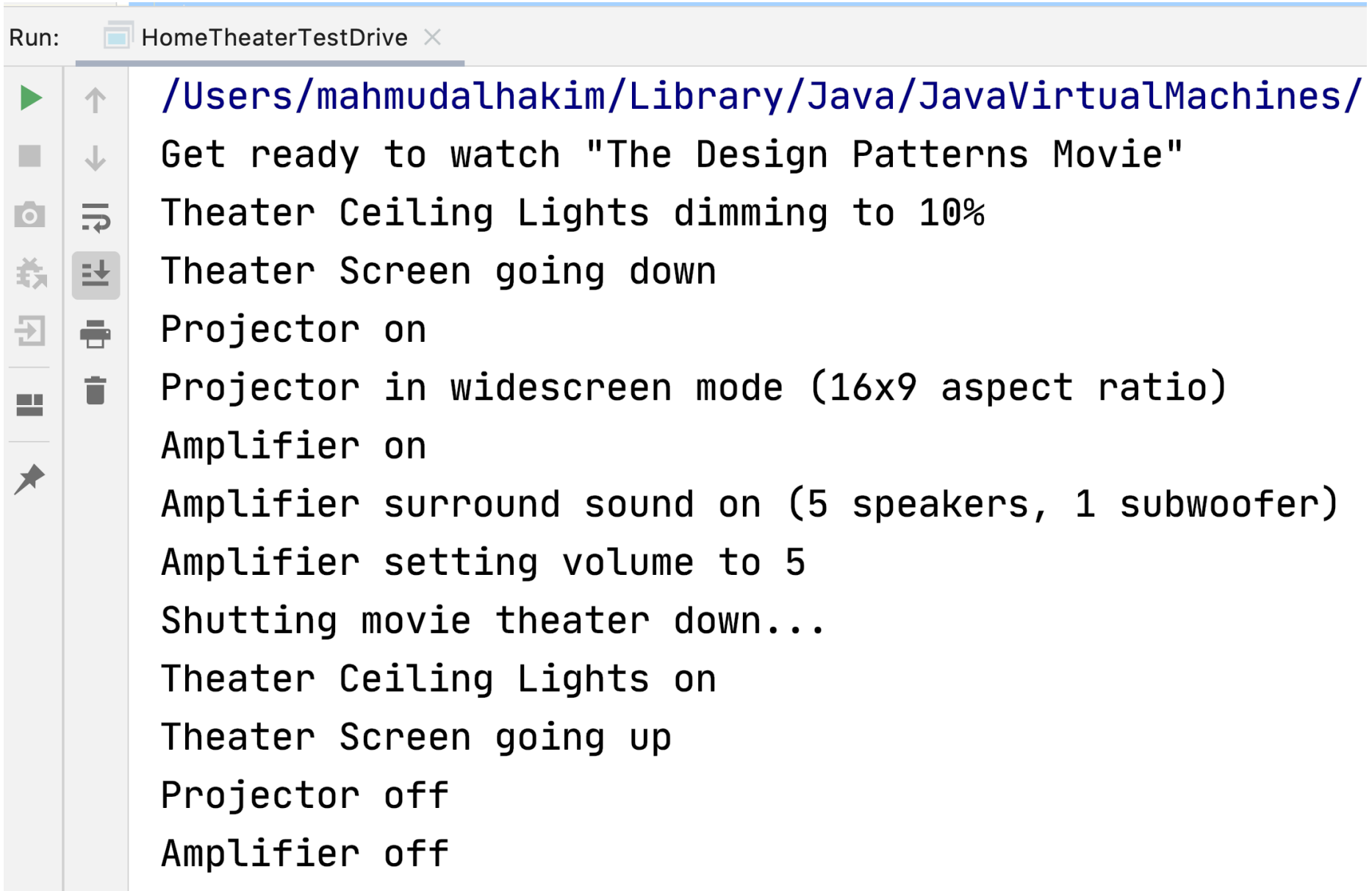
```
public static void main(String[] args) {

  Amplifier amp = new Amplifier("Amplifier");
  Projector projector = new Projector("Projector");
  TheaterLights lights = new TheaterLights("Theater Ceiling Lights");
  Screen screen = new Screen("Theater Screen");

  HomeTheaterFacade homeTheater =
      new HomeTheaterFacade(amp, projector, screen, lights);

  homeTheater.watchMovie("The Design Patterns Movie");
  homeTheater.endMovie();
}
```

Övning 4 – Exempel vid körning

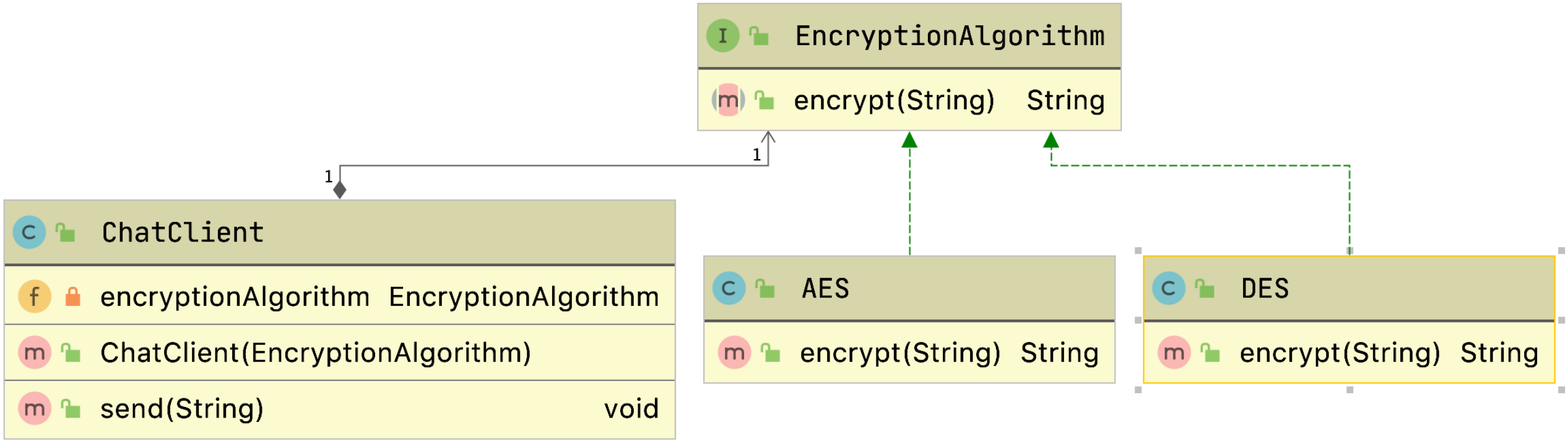


Övning 5: Refaktorera koden nedan med Strategy Pattern

```
public class ChatClient {
    private String encryptionAlgorithm;

    public ChatClient(String encryptionAlgorithm) {
        this.encryptionAlgorithm = encryptionAlgorithm;
    }
    public void send(String message) {
        if (encryptionAlgorithm == "DES")
            System.out.println("Encrypting message using DES");
        else if (encryptionAlgorithm == "AES")
            System.out.println("Encrypting message using AES");
        else
            throw new UnsupportedOperationException
            ("Unsupported encryption algorithm");
        System.out.println("Sending the encrypted message...");
    }
}
```

Övning 5: Class Diagram



Autogenererad klassdiagram från IntelliJ IDEA

Övning 6: Implementera Builder Pattern

```
// Without a Builder
Person p1 = new Person("Mahmud", 50, 85, 1.7);

// With a Builder
PersonBuilder p2 = new PersonBuilder()
    .setName("Mahmud")
    .setWeight(85)
    .setLength(1.7)
    .setAge(50);
```

Övning 7

- Klassen StringBuilder i Java använder ”Builder Pattern”.
- Gå igenom följande tutorial
  https://docs.oracle.com/javase/tutorial/java/data/buffers.html
- Skapa programmet **palindrome** * (Använd StringBuilder).
- Komplettera programmet genom att testa om en sträng är en palindrom.
  Testa t.ex. följande sträng: "Ni talar bra latin"
- Hur fungerar klassen StringBuilder?
  Vad är skillnaden jämfört med en vanlig String?

* En palindrom är en följd av skrivtecken som, med blanksteg och skiljetecken exkluderade, förblir oförändrad om man läser den baklänges.