Gestionarea unui cinema

Cărpinișan Sergiu Javier

1 Introducere

Proiectul realizat are ca utilitate gestionarea unui cinema, prin administrarea angajaților, clienților, sălilor de film și filmelor.

2 Laboratorul 1

2.1 Tipuri de valori

```
private String title;  4 usages
private int length;  4 usages
private Rating rating;  4 usages
```

2.2 Funcții

```
public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public int getLength() { return length; }
```

2.3 Output

```
System.out.println("Choose an option:");
System.out.println("1. Show all movies");
System.out.println("2. Show all cinema rooms");
System.out.println("3. Show all customers");
```

3 Laboratorul 2

3.1 While + switch

```
while (running) {
    int choice = scanner.nextInt();
    switch (choice) {
        case 1: {
            cinema.printMovies();
            break;
        }
        case 2: {
            cinema.printRooms();
            break;
        }
```

3.2 For

```
for (Room room : rooms) {
    System.out.println(room.toString());
}
```

3.3 If

```
if (customer.getId().equals(customerId)) {
    customers.remove(customer);
    System.out.println("Customer removed");
    break;
```

4 Laboratorul 3

4.1 ArrayList

```
private ArrayList<Customer> customers = new ArrayList<>();  7 usages
private ArrayList<Employee> employees = new ArrayList<>();  7 usages
```

4.2 Add

```
customers.add(new Customer(firstName, lastName, birthDate));
```

4.3 Remove

```
customers.remove(customer);
```

4.4 For-each loop

```
for (Customer customer : customers) {
```

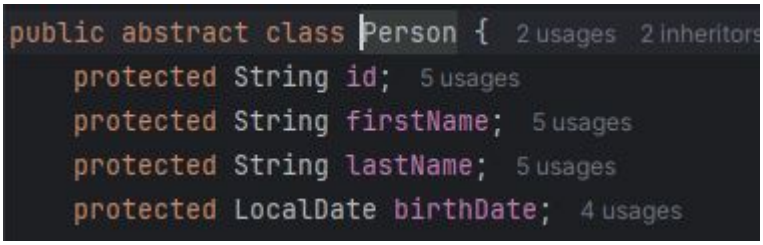
5 Laboratorul 4

5.1 Clasa cu atribute, metode de Get și Set

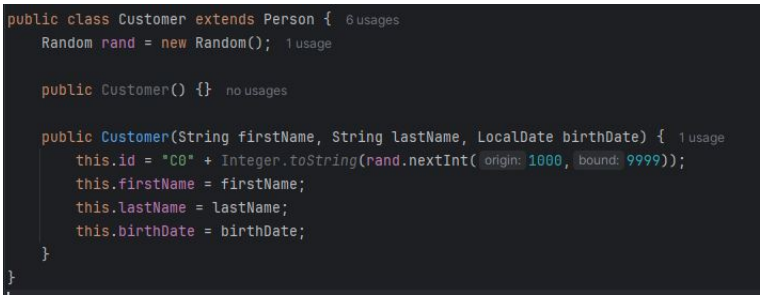
```
public abstract class Person {  2 usages  2 inheritors
    protected String id;  5 usages
    protected String firstName;  5 usages
    protected String lastName;  5 usages
    protected LocalDate birthDate;  4 usages

    public String getId() { return id; }

    public void setId(String id) { this.id = id; }

    public LocalDate getBirthDate() { return birthDate; }

    public void setBirthDate(LocalDate birthDate) { this.birthDate = birthDate; }

    public String getLastName() { return lastName; }

    public void setLastName(String lastName) { this.lastName = lastName; }

    public String getFirstName() { return firstName; }

    public void setFirstName(String firstName) { this.firstName = firstName; }

    @Override
    public String toString() {
        return "{" +
                "id='" + id + '\'' +
                ", firstName='" + firstName + '\'' +
                ", lastName='" + lastName + '\'' +
                '}';
    }
}
```

6 Laboratorul 5

6.1 Clasa abstractă

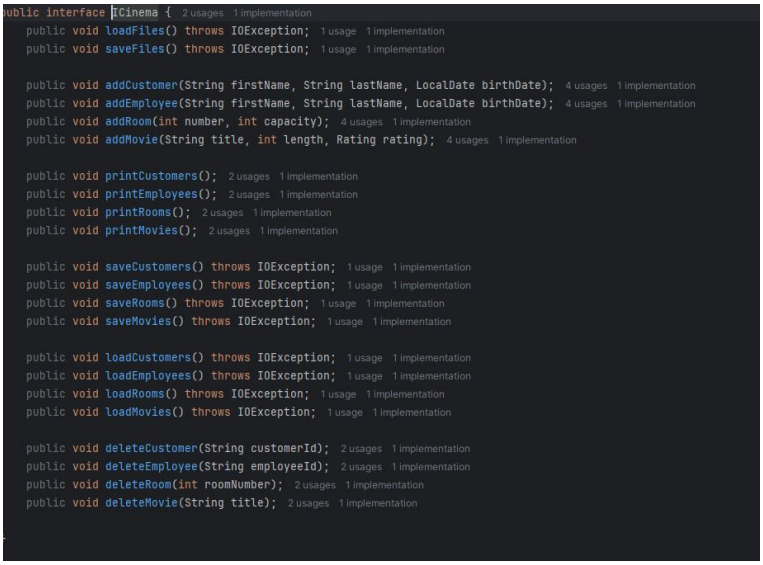


6.2 Clasa care extinde clasa abstractă



7 Laboratorul 6

7.1 Interfață



8 Laboratorul 7

8.1 Teste

```java
@Test
void testAddCustomer() {
    cinema.addCustomer( firstName: "John",  lastName: "Doe", LocalDate.of( year: 1990,  month: 1,  dayOfMonth: 1));
    assertEquals( expected: 1, cinema.getCustomers().size());
    assertEquals( expected: "John", cinema.getCustomers().get(0).getFirstName());
}

@Test
void testAddEmployee() {
    cinema.addEmployee( firstName: "Jane",  lastName: "Doe", LocalDate.of( year: 1985,  month: 5,  dayOfMonth: 20));
    assertEquals( expected: 1, cinema.getEmployees().size());
    assertEquals( expected: "Jane", cinema.getEmployees().get(0).getFirstName());
}

@Test
void testAddRoom() {
    cinema.addRoom( number: 101,  capacity: 50);
    assertEquals( expected: 1, cinema.getRooms().size());
    assertEquals( expected: 101, cinema.getRooms().get(0).getNumber());
}

@Test
void testAddMovie() {
    cinema.addMovie( title: "Inception",  length: 148, Rating.PG_13);
    assertEquals( expected: 1, cinema.getMovies().size());
    assertEquals( expected: "Inception", cinema.getMovies().get(0).getTitle());
}

@Test
void testDeleteCustomer() {
    cinema.addCustomer( firstName: "John",  lastName: "Doe", LocalDate.of( year: 1990,  month: 1,  dayOfMonth: 1));
    String customerId = cinema.getCustomers().get(0).getId();
```

9 Laboratorul 8

9.1 Stocare și încărcare date dintr-un fișier JSON

```java
public class JsonParser {  4 usages
    private ObjectMapper objectMapper = defaultObjectMapper();  3 usages

    private ObjectMapper defaultObjectMapper() {  1 usage
        ObjectMapper objectMapper = new ObjectMapper();
        objectMapper.registerModule(new JavaTimeModule());

        return objectMapper;
    }


    public <T> void writeListToFile(ArrayList<T> list, String filePath) throws IOException {  4 usages
        objectMapper.writerWithDefaultPrettyPrinter().writeValue(new File(filePath), list);
    }

    public <T> ArrayList<T> readListFromFile(Class<T> clazz, String filePath) throws IOException {  4 usages
        File file = new File(filePath);

        if (!file.exists()) {
            return new ArrayList<>();
        }

        return objectMapper.readValue(file, objectMapper.getTypeFactory().constructCollectionType(ArrayList.class, clazz));
    }
}
```

10 Laboratorul 9

10.1 Class diagram