Universitatea Dunărea de Jos

Facultatea de Automatică și Calculatoare

Proiect

Roboban - IP

Bogos Tudor

Grupa 22c32A

Îndrumător de proiect: Ş.l. dr. inf. Sabina COSTACHE

## Introducere

Acest eseu descrie un joc de tip „Roboban” (similar cu Sokoban) dezvoltat în Java, punând accent pe folosirea a **cinci șabloane de proiectare** (Singleton, Command, Adapter, Mediator, Builder). Proiectul este organizat în mai multe pachete, fiecare având o anumită responsabilitate. Documentul de față acoperă toate cerințele solicitate:

1. Dosar cu diagrama de clase și de pachete, evidențiind șabloanele, principiile și euristicile.
2. Calculul indicelui de stabilitate pentru fiecare pachet.
3. Folosirea a minim 5 șabloane de proiectare (cel puțin unul din fiecare categorie).
4. Evidențierea arhitecturii sistemului.
5. Plan de testare și lista cu cazuri de test + rezultatele testelor JUnit.

Pe parcurs, vor fi menționate clar locurile unde se vor insera **capturile de ecran** cu diagrame și structuri de pachete. Pentru o lectură mai facilă, se vor folosi secțiuni separate. Dimensiunea acestui eseu este una extinsă, pentru a detalia cât mai bine toate cerințele.

## 1. Cele 5 șabloane de proiectare folosite

În acest proiect s-au implementat cel puțin câte un șablon din fiecare categorie importantă, după cum urmează:

### 1.1. Creational – Builder (LevelBuilder)

* LevelBuilder construiește progresiv un Board de dimensiuni specificate, apoi adaugă un Player, una sau mai multe Box etc.
* Ușurează crearea de niveluri noi (nivel 1, nivel 2 etc.) cu parametri diferiți.

### 1.2. Structural – Adapter (AudioAdapter)

* AudioAdapter implementează interfața GameAudioInterface și folosește o librărie externă (ExternalSoundLib) pentru redarea fișierelor audio.
* Șablonul Adapter ascunde implementarea concretă de redare a fișierelor .wav.

### 1.3. Behavioral – Command (MoveCommand)

* Command definește semnătura execute().
* MoveCommand implementează logica de mutare a player-ului, împingerea cutiei și actualizarea pozițiilor. Face jocul extensibil: putem adăuga oricând UndoCommand sau alte tipuri de comenzi (ex. “JumpCommand”, “PullCommand” etc.).

### 1.4. Behavioral – Mediator (GameMediator)

* GameMediator coordonează interacțiunile dintre Board, Player și Box.
* Când GamePanel recepționează o tastă, apelează mediator.movePlayer(Direction). Mediatorul creează MoveCommand și îl execută.

### 1.5. Singleton – GameManager

* Gestionează starea globală (nivel curent, scor, mișcări).
* Are o metodă statică getInstance() care asigură existența unei singure instanțe la runtime.

## 2. Structura de clase și de pachete

### 2.1. Structura pachetelor

Proiectul este împărțit în mai multe pachete, fiecare având o responsabilitate specifică:

* **com.roboban.model**  
  Conține clasele de bază:
  + Board – Modelul matricii de joc (8x8).
  + Tile – Fiecărei poziții din board i se asociază un Tile cu proprietăți (walkable, goalTile).
  + Player – Modelul jucătorului, conținând coordonate (x,y).
  + Box – Modelul cutiei, împinsă de player.
  + Direction – Un enum simplu (UP, DOWN, LEFT, RIGHT).

A screenshot of a computer program

Description automatically generated

* **com.roboban.singleton**
  + GameManager – Clasa Singleton, responsabilă de gestionarea stării globale a jocului (nivel curent, scor, număr de mișcări).

A screenshot of a computer program

Description automatically generated

* **com.roboban.command**
  + Command – Interfața care definește metoda execute().
  + A screenshot of a computer

    Description automatically generatedMoveCommand – Implementarea concretă pentru mutarea player-ului și împingerea cutiilor.

A screenshot of a computer screen

Description automatically generated

* **com.roboban.mediator**
  + GameMediator – Centralizează interacțiunile dintre Board, Player și Box. Apelurile de mișcare se fac prin movePlayer(Direction).

A screen shot of a computer program

Description automatically generated

* **com.roboban.adapter**
  + GameAudioInterface – Interfață care definește metodele de redare/stoppare a muzicii.
  + ExternalSoundLib – Clasă care folosește AudioSystem (sau altă librărie) pentru a reda efectiv fișiere .wav.
  + AudioAdapter – Clasă care adaptează ExternalSoundLib la GameAudioInterface.

A screenshot of a computer program

Description automatically generated

* **com.roboban.builder**
  + LevelBuilder – Oferă metode de construire progresivă a obiectelor de joc (board, player, cutii) pentru un anumit nivel.

A screen shot of a computer program

Description automatically generated

* **com.roboban.ui**
  + GameWindow – Fereastra principală (un JFrame), conținând un CardLayout cu panouri distincte (MainMenu, Level1, Level2).
  + MainMenuPanel – Meniul principal (Start, Exit).
  + GamePanel – Panoul pentru fiecare nivel, gestionând interfața grafică și tastatura.

A screenshot of a computer program

Description automatically generated

## 2.2. Diagrama de clase

Următoarea diagramă ilustrează relațiile principale între clasele esențiale. Se pot sublinia cele cinci design pattern-uri după cum urmează:

1. **Singleton**: GameManager
2. **Command**: Command (interfață), MoveCommand (implementare)
3. **Adapter**: GameAudioInterface, ExternalSoundLib, AudioAdapter
4. **Mediator**: GameMediator
5. **Builder**: LevelBuilder

În diagrama de clase, se va vedea cum:

* GamePanel folosește un GameMediator (când se apasă o tastă, se apelează mediator.movePlayer).
* MoveCommand este construit în interiorul lui GameMediator și executat.
* Board conține Tile[][], un Player și o listă de Box.
* GameManager este o referință statică (unică).

A screenshot of a computer screen

Description automatically generated

## 3. Calculul indicelui de stabilitate pentru fiecare pachet

Pentru a evalua stabilitatea pachetelor, folosim formula simplă de la Robert C. Martin:

I= Fan-Out / ( Fan-Out + Fan-In)

unde:

* **Fan-In** = numărul de pachete care depind de acest pachet.
* **Fan-Out** = numărul de pachete de care depinde acest pachet.

Valoarea lui **I** variază între 0 și 1:

* **I=0** indică un pachet foarte stabil (are multe dependențe interne, iar alte pachete nu depind prea mult de el).
* **I=1** indică un pachet foarte instabil (depinde de multe alte pachete, dar nu este folosit aproape deloc de pachete externe).

|  |  |  |  |
| --- | --- | --- | --- |
| Package | Fan-In | Fan-Out | Stability Index |
| com.roboban.model | 8 | 0 | 0 |
| com.roboban.singleton | 3 | 0 | 0 |
| com.roboban.command | 2 | 0 | 0 |
| com.roboban.mediator | 1 | 4 | 0.8 |
| com.roboban.adapter | 2 | 0 | 0 |
| com.roboban.builder | 1 | 3 | 0.75 |
| com.roboban.ui | 0 | 10 | 1 |

## 4. Arhitectura sistemului

Sistemul este organizat pe **straturi**:

1. **Prezentare / UI** (com.roboban.ui):
   * Conține GameWindow, MainMenuPanel, GamePanel.
   * Interacționează cu GameMediator pentru logica de mutare și cu GameManager pentru scor/nivel.
   * Desenează board-ul (metoda paintComponent), afișează butoane și gestionarea tastelor.
2. **Logica** (pattern-urile command, mediator, builder, plus modelul Board):
   * GameMediator centralizează mutările player-ului.
   * MoveCommand execută logica efectivă, manipulând instanțele din Board.
   * LevelBuilder creează structura de date pentru nivelurile de joc.
3. **Infrastructură** (adapter, singleton):
   * AudioAdapter se ocupă de muzică.
   * GameManager (Singleton) reține starea globală, accesibilă de oriunde (nivel, scor etc.).

Această separare permite extinderea jocului prin adăugarea mai multor niveluri, a altor tipuri de entități (ex. inamici), a altor tipuri de comenzi (undo, pauză) ș.a.m.d.

## 5. Plan de testare, lista cu cazuri de test și rezultate

### 5.1. Plan de testare

Obiectivul testării este să validăm:

1. **Funcționalitatea de bază** (modelul: Board, Player, Box).
2. **Interacțiunile** (mediator, command).
3. **Starea globală** (singleton).
4. **Inițializarea** (builder).
5. **Redarea muzicii** (adapter) – la nivel minimal, putem verifica să nu existe excepții la apelurile de redare.

### 5.2. Lista cu cazuri de test

* **BoardTest**
  + Verificarea dimensiunilor (pozitii valide/invalide).
  + Adăugarea Player-ului, cutiei, tile walkable/goalTile.
* **EntitiesTest** (pentru Player și Box)
  + Poziționare inițială, setare x,y.
* **MoveCommandTest**
  + Mutarea liberă a player-ului.
  + Împingerea unei cutii dacă există spațiu.
  + Blocarea mișcării dacă nu e spațiu.
* **GameMediatorTest**
  + Integrarea dintre Board și MoveCommand: movePlayer(Direction.X).
  + Verificarea reacției la obstacole.
* **GameManagerTest**
  + Verificăm existența unică (Singleton).
  + Gestionează corect currentLevel, score, movesCount.
* **LevelBuilderTest**
  + Crearea unui Board cu dimensiuni, player și cutii setate.A green rectangular object with a black border

    Description automatically generated

**5.3. Rezultatele testelor cu JUnit**

Toate testele JUnit se pot rula fie din IDE (click dreapta → Run Tests), fie cu mvn test (dacă se folosește Maven). În urma rulării testelor, se vor obține rapoarte de tip:

* **Rezultate**:
  + BoardTest – Passed: 4/4
  + EntitiesTest – Passed: 2/2
  + MoveCommandTest – Passed: 3/3
  + GameMediatorTest – Passed: 2/2
  + GameManagerTest – Passed: 3/3
  + A screenshot of a computer program

    Description automatically generatedLevelBuilderTest – Passed: 1/1