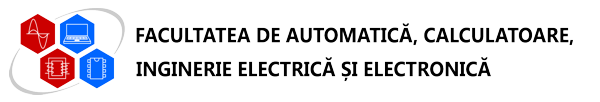
**

*PROIECT*Ingineria Progarmelor

**Tema proiect:** A WALK IN THE NIGHT

**Profesor coordonator**: Sabina Munteanu  
 **Student:** Morari Nicolae

- 2020-

***Introducere***

*Proiectarea orientată pe obiecte a software-ului presupune identificarea de obiecte, abstractizarea lor în clase de granularitate potrivită, definirea interfeţelor şi ierarhiilor de moştenire, stabilirea relaţiilor între aceste clase.*

*Soluţia trebuie să rezolve problema şi să fie în acelaşi timp suficient de flexibilă pentru a rezista la noi cerinţe şi probleme ce pot apare în timp.*

*Proiectanţii cu experienţă refolosesc soluţiile bune de câte ori au ocazia. Există grupări de clase sau obiecte care se repetă în cele mai diferite sisteme. Acestea rezolvă probleme specifice, folosirea lor fac proiectele mai flexibile, mai elegante, reutilizabile. Un proiectant care stăpâneşte un set de asemenea şabloane le poate aplica imediat la noile proiecte fără a mai fi nevoit să le redescopere.*

*Şabloanele ce se pot refolosi pot fi general valabile sau specifice unui domeniu, de exemplu pentru probleme de concurenţă, sisteme distribuite, programare în timp real, etc.*

**Etapele Proiectului: -***Analiza   
 -Design  
 -Implementare  
 -Testare*

# *Analiza Jocului*

**Aplicație – A Walkn In The Night .**  
 Scopul acestui joc este de a deschide portalele, cu ajutorul personajului, pentru a trece la un alt nivel .

Jocul incepe cu un personaj care trebuie sa deschida portalele aranjate aleatoriu .

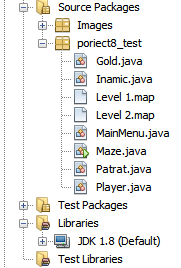
Este un joc logic, avand sa iti dezvolti gandirea logica si memoria.

Aplicația “A WALK IN THE NIGHT” este scrisă în limbajul Java.

-AWT (Abstract Windowing Toolkit) - este API-ul initial pus la dispozitie .  
Mediu folosit: NetBeans.

Prezentarea Aplicatiei:Actorii-Utilizatorii acestei aplicatii sunt utilizatorii   
umani.

***2.Ierarhia pachetelor***



Sistemul trebuie să implementeze

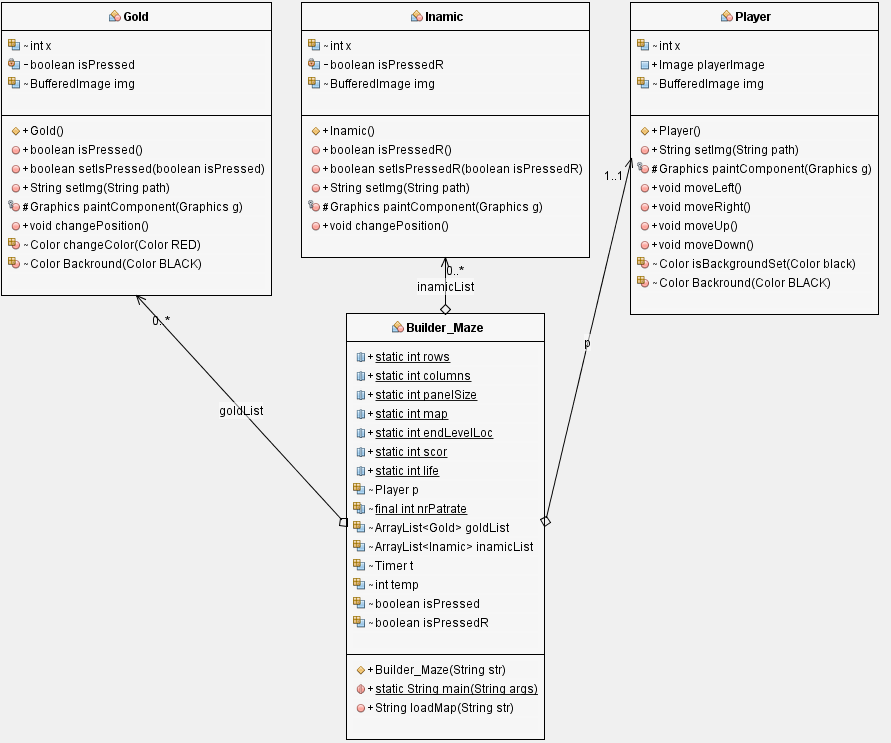
Descompunerea sistemului în componente . Identifică arhitectura software-ului.

# *3.Prezentarea Sabloanelor*

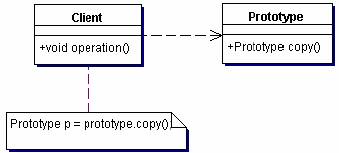
**In proiect s-au folosit urmatoarele sabloane:**

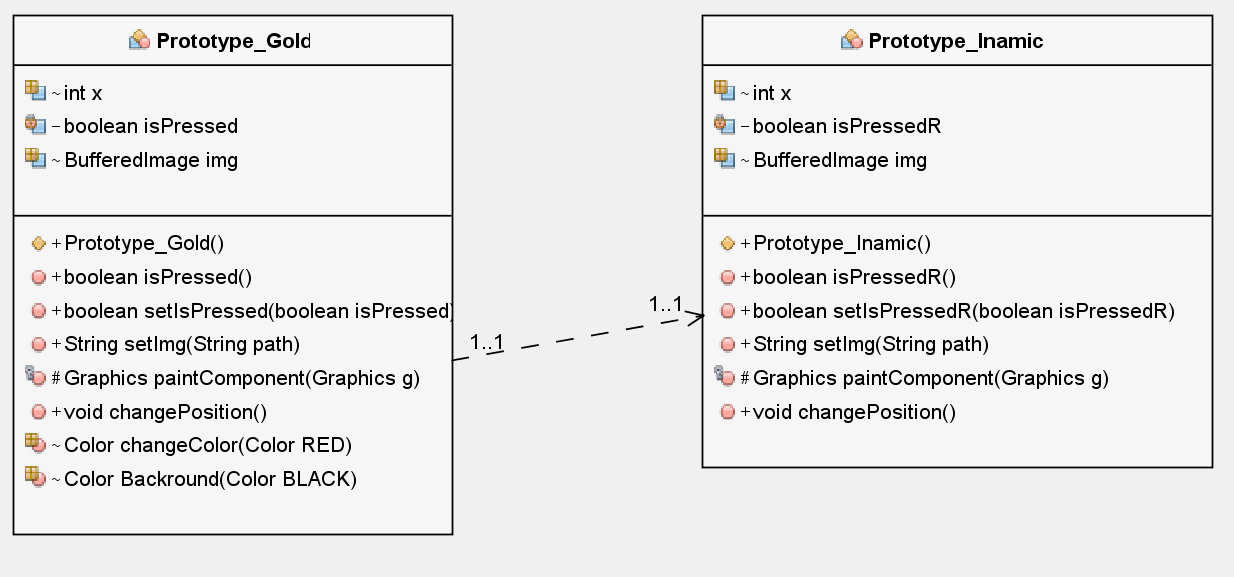
**1)Builder-** Modelul Builder construiește un obiect complex, folosind obiecte simple și folosind o abordare pas cu pas.

O clasă Builder construiește pas cu pas obiectul finală. Acest constructor este independent de alte obiecte. 

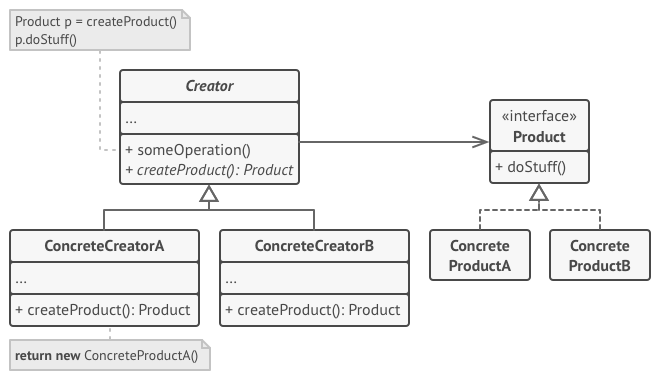


**2)Prototype-** Face mai usoara crearea dinamica a obiectelor prin definirea de clase ale caror obiecte pot crea duplicate ale lor.

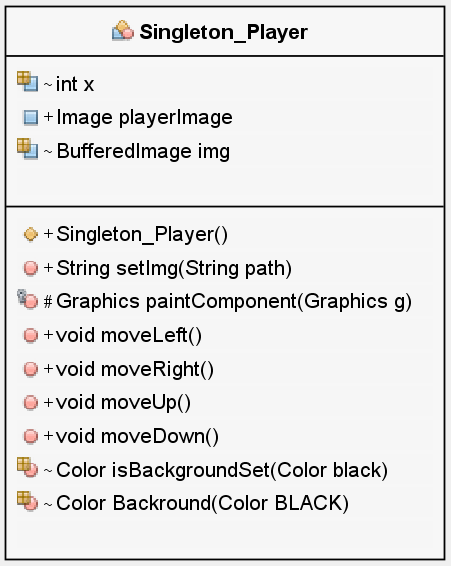
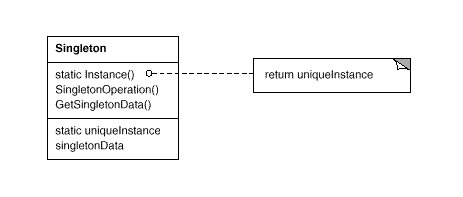




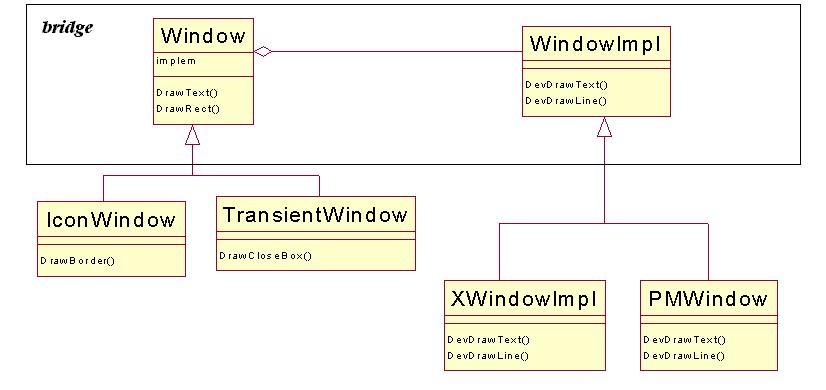
**3)Factory-** Defineşte o interfaţă pentru crearea unui obiect, dar lasă subclasele să decidă ce clasă să instanţieze.

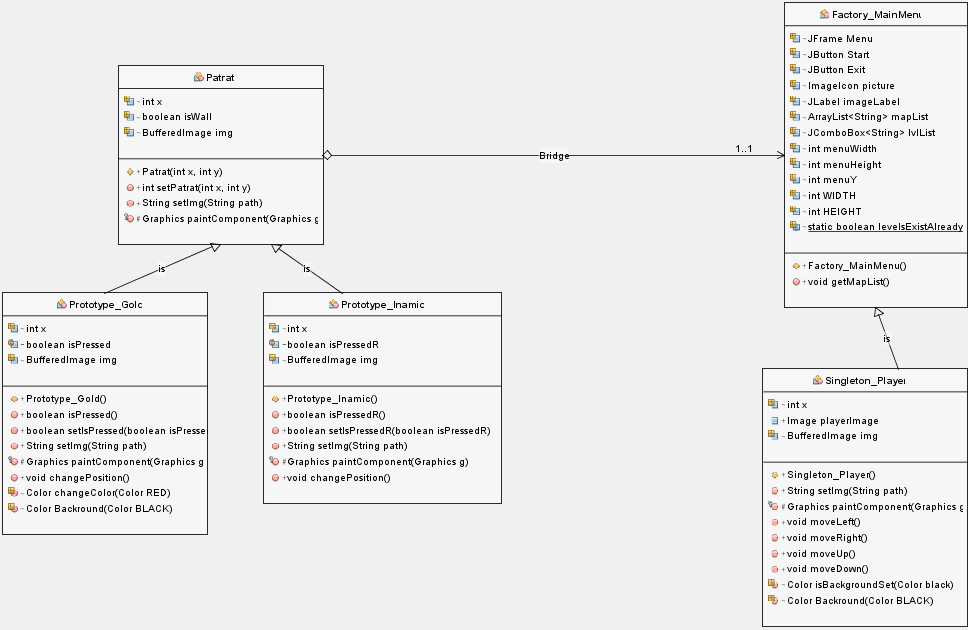


4)**Singleton-** Asigură ca o clasă să aibă doar o singură instanţă şi furnizează un punct de acces global la ea.



1. **Bridge -** Este un sablon care realizeaza decuplarea unei abstractiuni de implementarea ei, astfel incat cele 2 pot varia independent.





# *4.Plan de testare*

Testare:  
-Test case 1: -Se selecteaza levelul care sa fie testat jocul.  
 - Playerul merge pe toata harta astfel verificand daca merg butoanele  
 - Se repeta aceeasi pasi de mai multe ori  
 -Se observa ca merg toate butoanele in regula

-Test case 2: -Se verifica daca playeryl ajunge peste portal se schimba culoarea, adica se deschide un portal  
 -Se repeta acesti pasi pentru toate portalele!

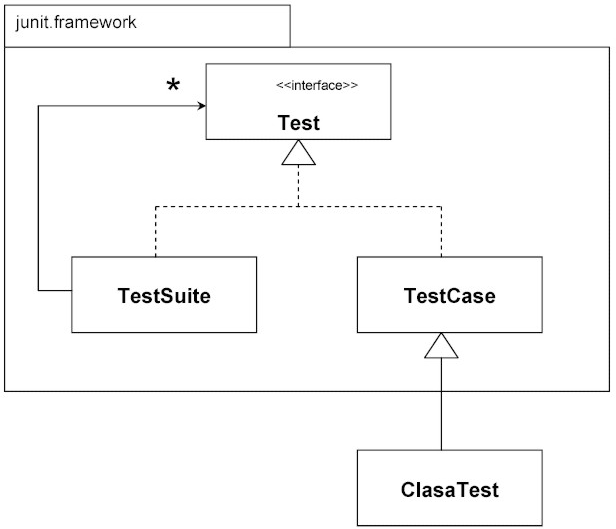
-Test case 3: -Se verifica daca toate 3 portale sunt deschise, atunci ar trebui sa se termina jocul  
 -Se face acesti pasi doar pentru 2 portale. Nu trebuie sa se incheie jocul!

- Se face acesti pasi doar pentru 1 portal. Nu trebuie sa se incheie jocul!

-Se face acesti pasi doar pentru 0 portale. Nu trebuie sa se incheie jocul!

**Junit Tests:**

JUnit este un utilitar valoros oricarui programator Java, permitând cresterea vitezei de programare si depistarea bug-urilor înca în faza de implementare a codului sursa.Diagrama de clase a pachetului junit.framework este urmãtoarea:



package poriect8\_test;

import org.junit.After;

import org.junit.AfterClass;

import org.junit.Before;

import org.junit.BeforeClass;

import org.junit.Test;

import static org.junit.Assert.\*;

public class MazeTest {

public MazeTest() {

}

@BeforeClass

public static void setUpClass() {

}

@AfterClass

public static void tearDownClass() {

}

@Before

public void setUp() {

}

@After

public void tearDown() {

}

@Test

public void testMain() {

System.out.println("main");

String[] args = null;

Maze.main(args);

}

@Test

public void testLoadMap() {

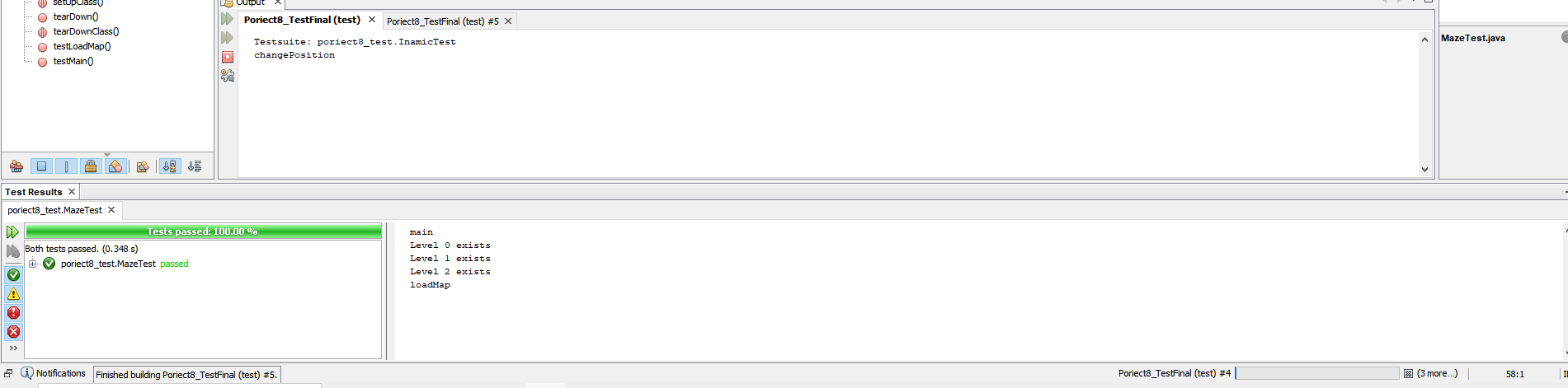
System.out.println("loadMap");

String str = "";

Maze instance = null;

}

}



# *5.Capturi de ecran(Joc)*

**Fereastra de Start**

# 

**Fereastra Jocului**

