*Motivarea baremului de notare pentru evaluarea proiectului*

*Proiectul “ChadRobot” doreste sa fie un joc de dimensiuni mici, plecand de la premiza urmatoare: un robotel defect se afla intr-un mediu necunscut, inconjurat de desert. Directiva principala a robotelului? Colecteaza cat mai multe bijuterii, dar mai presus de atat, gaseste un cufar. Circumstantele fac insa ca, robotelul este defect, el pierzand combustibil la fiecare pas, iar receptoarele, compromise si acestea, distortioneaza lumea.*

*In randurile urmatoare, voi prezenta apartenenta cunostiintelor dobandite la laboratorul de MIP, conform baremului propus.*

***Lab1: Introducere în Java (output, tipuri de valori, funcții)***

Fiecare componenta adusa in discutie de acest proiect, este alcatuita din functii si variabile. Spre exemplu, in clasa AbstractRobot intalnim variabile de tip int (int xPos, yPos), boolean (boolean won). Iar din punctul de vedere al functiilor, se regasesc metode cu parametru, precum public void decreaseFuel(int value), sau fara parametru, public int getX(). Output-ul este gestionat de interfata grafica realizata cu ajutorul la Java Swing.

***Lab2: Introducere în Java (input, for, while, switch, if)***

Pentru a motiva existenta comenzilor evidentiate mai sus, propun urmatoarele functii:

-in functia public void processInput(char input) din GameManager, identificam un switch-case si un if-statement

-in functia public void moveLogic(int x, int y) intalnim if-statement in if-statement

-input-ul are loc atat de la tastatura, prin intermediul functiei public void keyPressed(KeyEvent e) din clasa Menu, dar si de la mouse, cu ajutorul butoanelor, din aceiasi clasa.

***Lab3: Colecții Java (Array, List, Map)***

Din punctul de vedere al structurilor de date, proiectul prezinta in principal, o lista, private List<List<Integer>> board in clasa Board, si un HashMap, private HashMap<String, Image> imageMap, in clasa ImageHash. Ca interfata pentru lista, am ales sa folosesc un ArrayList, in detrimentul la LinkedList.

Am ales sa folosesc un HashMap in clasa ImageHash pentru a memora eficient imaginile de care este nevoie la partea grafica a proiectului. Dintre functiile oferite de aceasta structura de date, am utilizat functia put() (imageMap.put(name, image)) si functia get() (imageMap.get(name)).

Din catalogul arraylist-ului, am folosit functia add() (board.add(0, generateLine()), functiile removeFirst() si removeLast() (board.removeFirst() si board.removeLast()), functiile de get() si set() (board.get(robotX).set(robotY, 0))).

***Lab4: Clase Java (clasă cu atribute și metode)***

Proiectul prezentat este structurat in conformitate cu paradigma obiect-orientata propusa de limabjul Java. Clasele pe care le-am creat sunt urmatoarele: Menu, ImageHash, Board, AbstractRobot, NormRobot, BuffRobot si GameManager. Fiecare clasa enumerata detine atribute, spre exemplu luam clasa GameManager, ce contie atribute precum private int score, fuel = 20 sau private static final int *boardSize* = 7 si metode specifice, cum ar fi public void loadScoreFromFile(String fileName), public void dumpScoreOnFile(String fileName) sau public void moveLogic(int x, int y). Toti parametrii si toate metodele au ca prefix specificatorul de acces din clasa respectiva. Totodata, am folosit si initalizarea unor parametrii prin intermediul operatorului =.

In clasa abstract AbstractRobot, intalnim si membrii protejati, cum ar fi protected int fuel, currentCoins.

Fiecare clasa detine si cate un constructor, pe exemplul propus mai sus public GameManager(), iar clasa Menu contine o functie main aditionala.

***Lab5: Moștenire în Java, clase abstracte***

Necesitatea existentei mostenirii si implicit a unei clase abstracte se regaseste cand discutam despre cei doi robotei oferiti spre a reprezenta player-ul. Jocul ofera alegerea intre un robot normal, NormRobot, si un robot “musculor”, BuffRobot. Ambii roboti sunt mosteniti din clasa abstracta AbstractRobot (declarata astfel: public abstract class AbstractRobot, utilizand cuvantul rezervat abstract). Clasa AbstractRobot permite existenta unor atribute si implementarea unor metode comune intre cei doi robotei, cum ar fi protected int xPos, yPos sau public void move(int x, int y). Robotul normal poate fi considerata ca o instantiere a clasei abstracte pe care o mosteneste, intrucat nu adauga niciun parametru sau metoda noua. In schimb, clasa BuffRobot include atat un parametru nou de tip boolean ( private boolean secondChange), cat si mecanici/metode noi ( public void useSecondChance() etc.). Ambele clase apeleaza in constructorul propriu constructorul clasei parinte, cu ajutorul functiei super (super(xPos, yPos, fuel)). De specificat este si faptul ca mostenirea se face evidenta prin adaugarea secventei “extends AbstractRobot”, cum se poate observa in implementarea clasei BuffRobot: public class BuffRobot extends AbstractRobot.

***Lab6: Interfețe în Java***

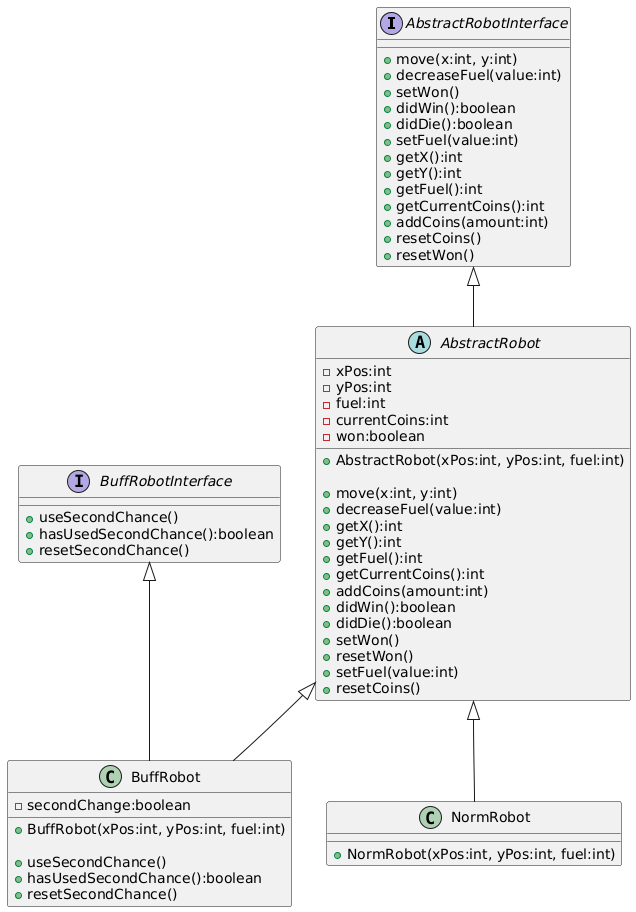
Referitor la roboteii mentionati anterior, asupra clasei abstracte am imprimat si o interfata, AbstractRobotInterface, ce cuprinde antetul functiilor implementate in clasa AbstractRobot. Astfel, forma clasei intr-un final este public abstract class AbstractRobot implements AbstractRobotInterface, folosind cuvantul rezervat implements. Totodata, am adaugat o interfata si pentru BuffRobot. Spre deosebire de clasele de pana acum, interfetele se folosesc de cuvantul “interface” in loc de “class”: public interface AbstractRobotInterface. In clasele respectiva, semnalarea ca are loc o implementare a unei functii din interfata are loc cu secventa “@Override” amplasata imediat inaintea declararii functiei in clase in care are loc implementarea.

***Lab7: Teste***

Intr-un package complet nou am aduagat teste pentru AbstractRobotInterface si pentru Board. In aceste clase menite pentru a testa anumite functii, regasim void decreaseFuel(), void setWon() , void didWin(), fiecare fiind precedate de secventa “@Test”. Totodata, pentru teste se utilizeaza, in loc de “return” functii de comparare si validare a raspunsurilor, precum *assertEqual, assertTrue sau assertFalse.*

***Lab8: Persistența datelor***

Pentru a memora scorul acumulat de un jucator de-a lungul a mai multe jocuri, am aduagat un fisier text, Score.txt. In momentul in care este apelat constructorul unui GameManager, atunci este apelata functia public void loadScoreFromFile(String fileName), care citeste din fisierul mentionat anterior. In urma completarii unei victorii, are loc apelarea functiei public void dumpScoreOnFile(String fileName), prin intermediul careia are loc inlocuirea vechiului scor din fisier cu noul scor. Functia de citire se foloseste de un BufferedReader si de un FileReader (BufferedReader br = new BufferedReader(new FileReader(fileName))), iar functia de scriere in fisier de un BufferedWriter si un FileWriter (BufferedWriter writer = new BufferedWriter(new FileWriter(fileName))).

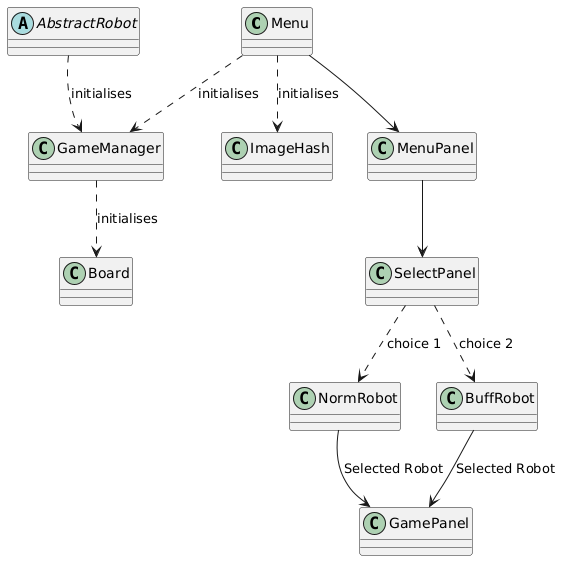
***Lab9: Realizați 3 diagrame UML***

Prima diagrama UML afiseaza schema mostenirilor din cadrul proiectului. Se poate observa cu usurinta ca AbstractRobot implementeaza interfata AbstractRobotInterface. La randul lor, NormRobot si BuffRobot extind clasa parinte, mostenindu-i atributele si metodele. Intr-un final, clasa BuffRobot implementeaza inca o interfata, BuffRobotInterface.

A computer screen shot of a computer

Description automatically generated with medium confidence

In a doua diagrama, se poate observa relatia dintre clasele principale ce apar in proiectul de fata. Fiecare celula contine tipul clasei din care face parte (clasa, interfata, clasa abstracta), numele clasei respective, atributele si metodele, care pot fi private, notate cu un patrat rosu, publice, notate cu un cerc verde sau protejate, notate cu un romb portocaliu. Pentru simplitatea diagramei, am omis adaugarea elementelor prezentate in prima diagrama.



Ultima diagrama ilustreaza Game-Flow-ul proiectului. Interactiunea jucatorului cu programul incepe din clasa Menu, continuand in MenuPanel. In spate, sunt initializate clasele GameManager si ImageHash. Jucatorul este nevoit sa aleaga intre cele doua variante de robotei, finalizand initializarea clasei GameManager. Intr-un final ajunge la GamePanel, loc unde are loc jocul, ilustriand board-ul si robotelul ales.