**Blossom Boy:A Mother’s Gift**

**Contextul jocului:**

Tim era un tânăr care trăia într-o lume în care natura era rară și prețuită. Mama lui era o cercetătoare faimoasă în domeniul botanic, iar Tim îi împărtășea pasiunea pentru plante. Într-o zi, mama lui i-a vorbit despre o floare rară, care creștea într-o curte părăsită din apropiere, dar care era păzită de goblini. Tim a decis să își facă curaj și să încerce să obțină acele flori pentru mama sa. A plecat în căutarea curții și, când a ajuns acolo, a descoperit că goblinii erau mai răi decât se așteptase.

Cu flori în mână, Tim s-a îndreptat spre ieșirea din curte. Însă goblinii erau la pândă și au încercat să îl prindă. Tim a dat tot ce a putut și a reușit să scape cu floarea în siguranță. Ajuns acasă, a oferit mama sa floarea dorită, făcând-o foarte fericită și mândră de fiul său curajos.

**Sistemul de joc:**

Jocul este tile-based cu vedere de sus în care jucătorul parcurge 3 nivele. Fiecare nivel prezinta un număr diferit de inimici cu puteri diferite, iar jucătorul trebuie sa îi evite colectand în același timp florile necesare pentru a trece la nivelul urmator. Deși jocul pare destul de simplu jucătorul va fi pus în dificultate din cauza faptului că fiecare nivel vine cu un număr diferit de inamici iar fiecare inamic are atacul lui special, astfel incat la ultimul nivel jucătorul va fi nevoit sa înfrunte toate cele 3 categorii de inamici cu atacuri diferite fiecare și cu un timp cat mai redus.

**Continutul jocului:**

-Monstri: goblinii care evoluează de la un nivel la altul și care se plimba pe harta pe tot parcursul nivelului, iar pe fiecare nivel avand atacuri diferite.Jocul contine atat goblini care ataca de aproape cât și de la distanța avand arme.

       -Eroul : Caracterul pe care jucătorul îl poate controla și care poate ajunge la viteze diferite în funcție de timpul în care colecteaza florile. Dacă acesta colectează prima floare într-un timp bun viteza eroului va crește puțin, însă dacă prima floare este pierdută viteza de deplasare va scădea. În plus eroul deține și o super putere aceea de alerga rapid pentru o perioada scurta(dash) dar cu un cooldown de cateva secunde între ele.

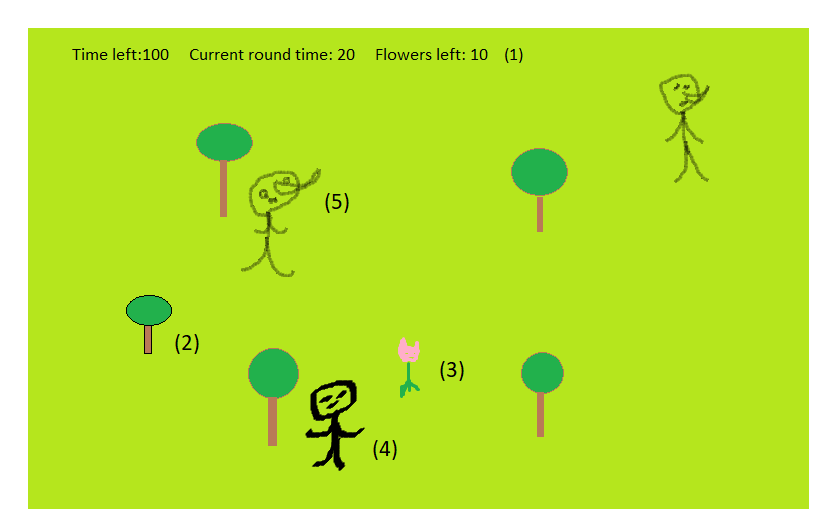
-Florile: sunt elementele pe care jucătorul trebuie sa le colecteze într-un timp bun și ferindu-se de inamici, acestea dandu-si respawn random pe harta de fiecare data cand sunt colectate.

**Niveluri**:

-**Nivelul 1:** inamicii doar se plimba pe harta fiind alertați de prezenta jucătorului doar dacă acesta trece foarte aproape de ei iar dacă aceștia te ating nivelul se reseteaza. De asemenea exista și un timp per total dar și un timp pentru fiecare floare în parte.

       -**Nivelul 2:** aici vor fi doua categorii de inamici: cei simpli de la primul nivel si cei care pot sa atace cu o sabie de la o anumită distanță. De asemenea timpul pentru colectarea florilor va fi scurtat.

       -**Nivelul 3:** în acest nivel vor fi numeroși inamici din nivelul unu și doi dar pe langa acestea vor apărea și inamicii specifici nivelului trei, și ce au ei nou este faptul că pot sa tragă cu pușca către erou. Timpul va fi destul de aspru si în defavoarea eroului dar jucătorul trebuie sa bage de seama ca acesta este ultimul nivel în încercarea de a termina jocul.



(1): Cele 3 bari  : - Prima reprezintă timpul total pentru a aduna toate florile.

        -A doua reprezinta timpul de colectare ramas pentru fiecare floare în parte.

      -A treia bara reprezinta numarul de flori ramase necolectate.

(2) :Copacii care diferă ca poziționare de la un nivel la altul și au rolul de a ascunde jucătorul cand este urmărit

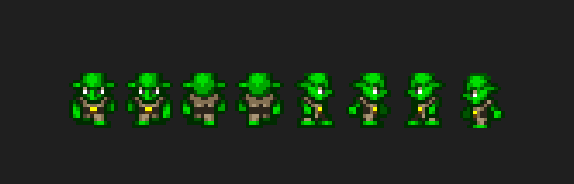
(3): Floarea care trebuie adunata de către jucător înainte de a se scurge timpul.

(4): Eroul pe care jucătorul îl controlează.

(5): Goblinii care pazesc florile și care-l împiedica pe jucător în încercarea lui de a termina jocul.

**Sprite-uri folosite**:

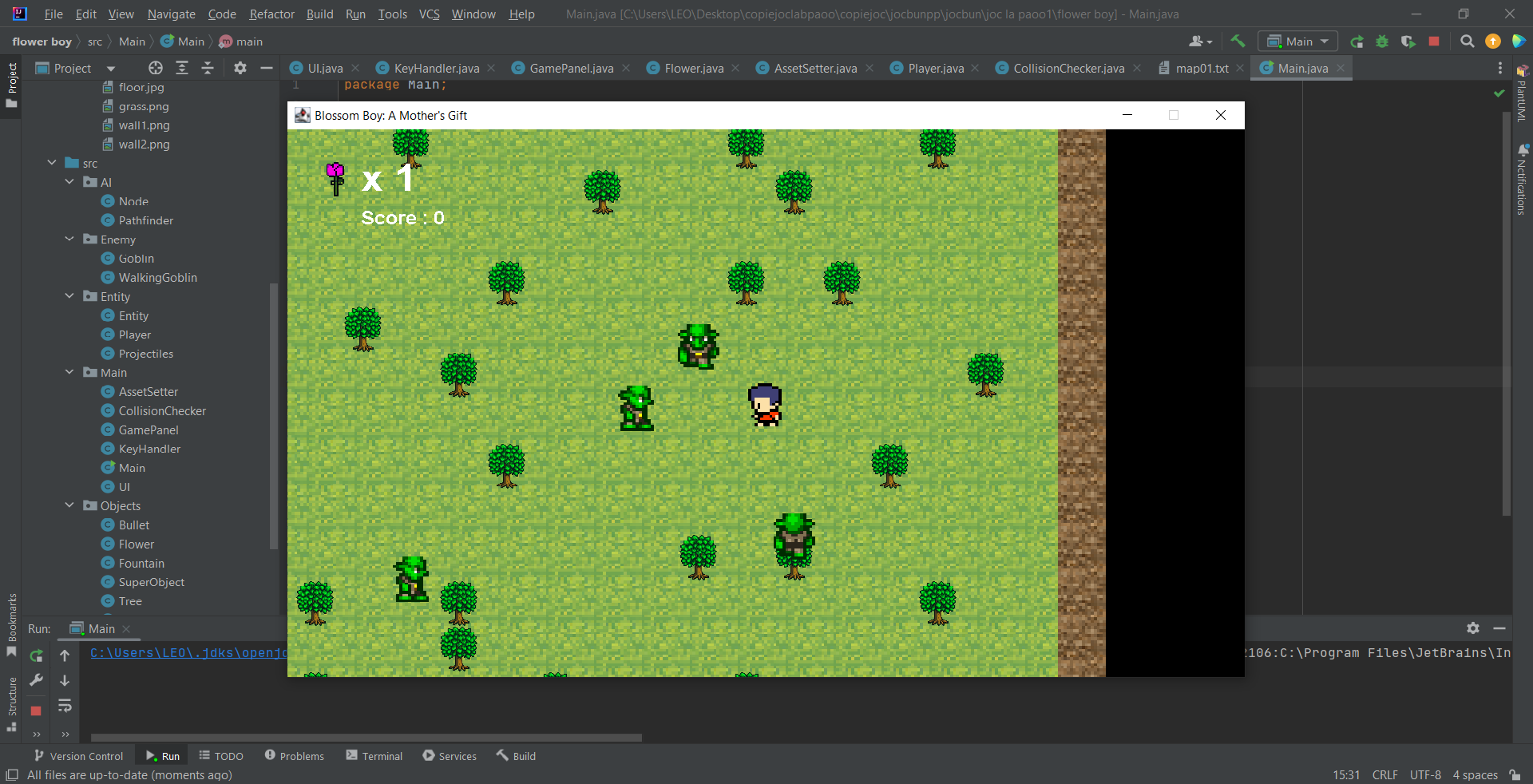
  grass - Copy.png 



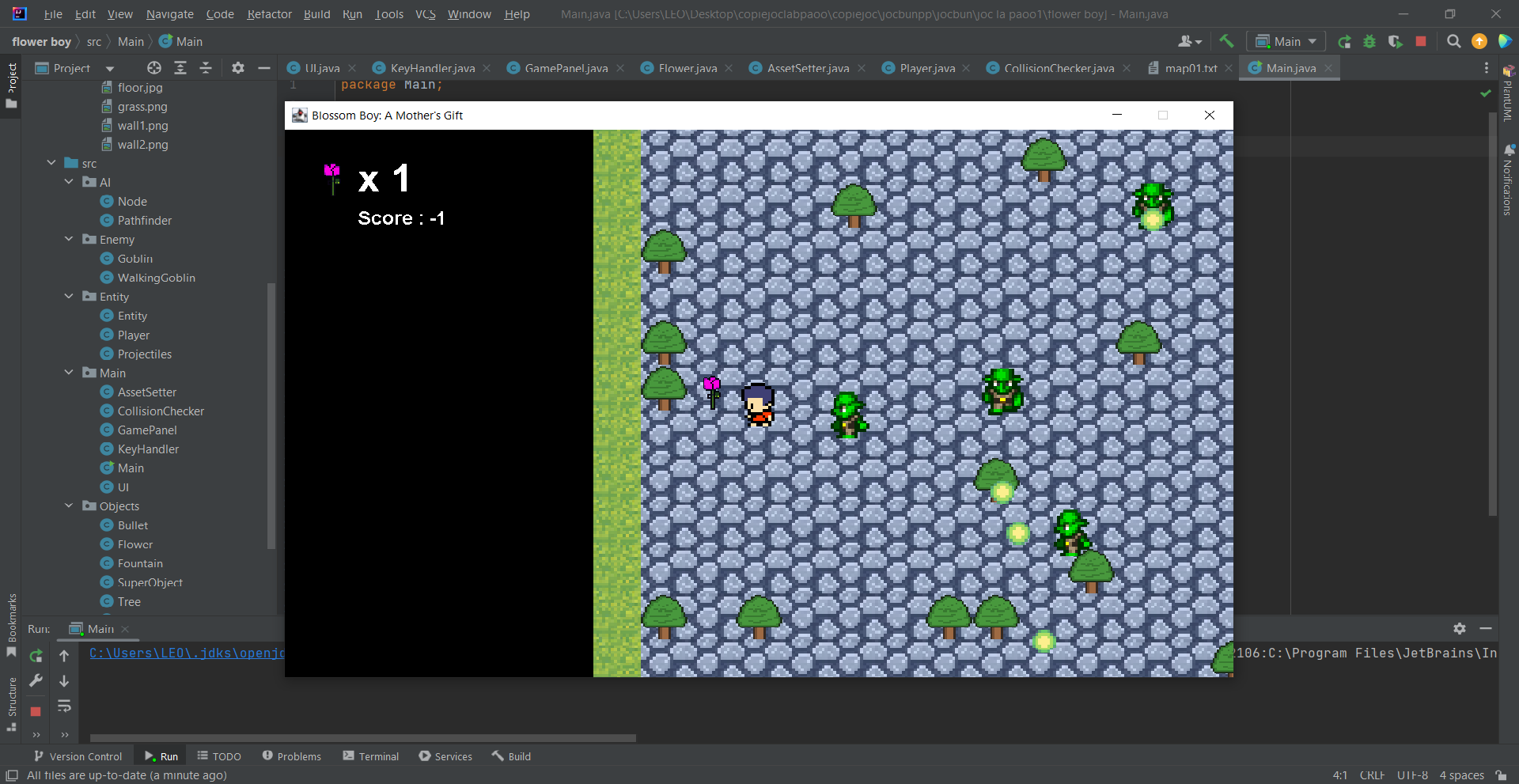


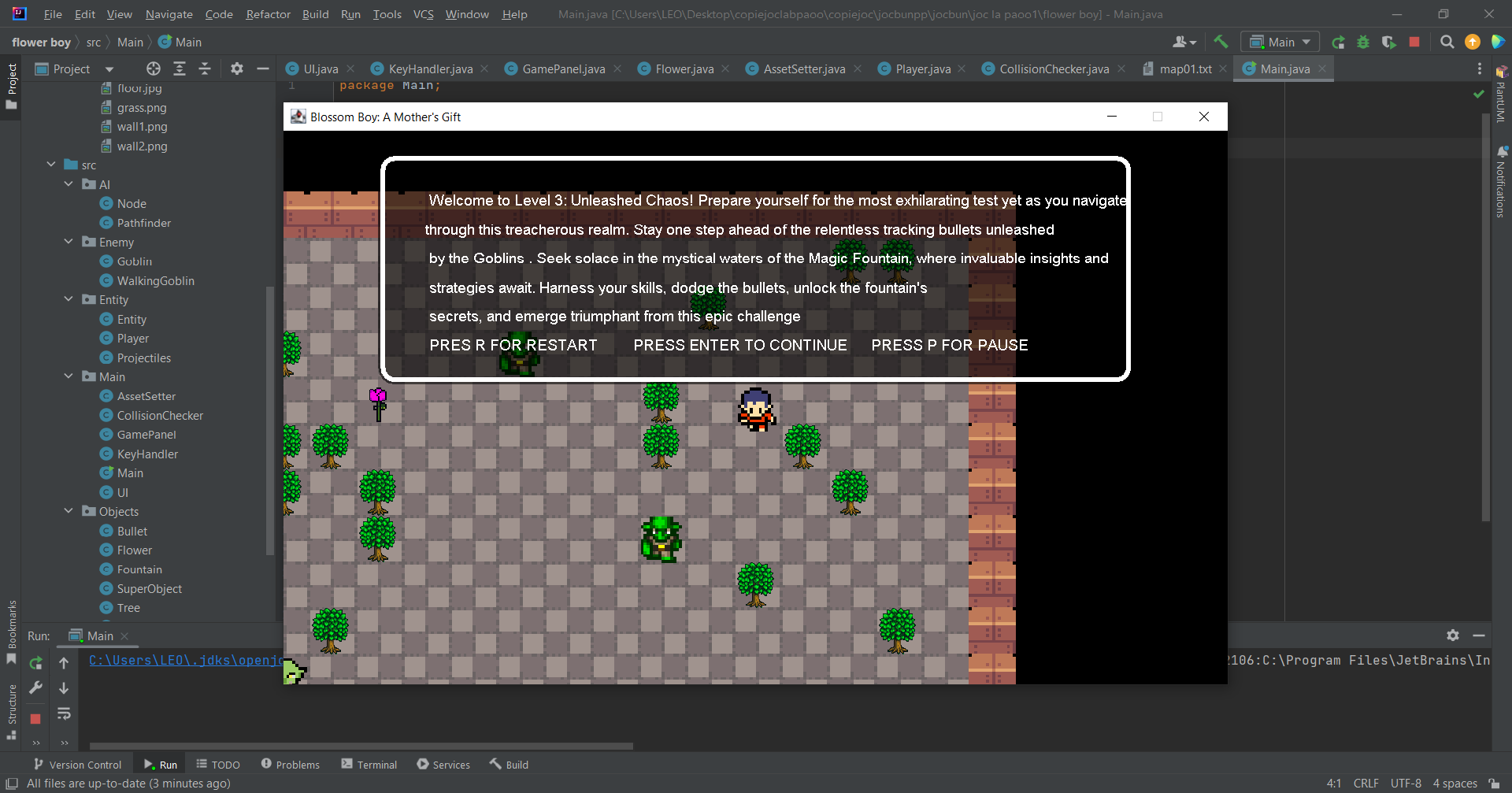
**Resurse biografice utilizate**:Sprite-urile au fost luate de pe : <https://itch.io/game-assets/tag-sprites>

**-Nivelul 1:**



**-Nivelul 2:**





**Main.java** - Aceasta este clasa principală pentru joc, creează o fereastră JFrame, adaugă un GamePanel la ea și setează titlul ferestrei.

**GamePanel.java** - Aceasta este clasa Game Panel care utilizează biblioteca Swing pentru a crea o interfață grafică pentru utilizator (GUI) pentru un joc. GUI-ul constă într-un panou numit GamePanel, care extinde clasa JPanel și conține diverse elemente precum tileuri, entități și un jucător.

**KeyHandler.java** - Clasa KeyHandler implementează interfața KeyListener și gestionează evenimentele tastaturii pentru un joc. Clasa KeyHandler are un obiect GamePanel, care este utilizat pentru a schimba starea jocului și pentru a reda sunete. Clasa are, de asemenea, variabile booleane pentru fiecare cheie utilizată în joc, care sunt setate la true atunci când tasta este apăsată și false când tasta este eliberată. În metoda keyPressed(), codul verifică care tastă este apăsată și setează variabila booleană corespunzătoare la true. În funcție de starea jocului, diferite acțiuni sunt realizate atunci când anumite taste sunt apăsate.

**UI.java** - Clasa UI este responsabilă de desenarea interfeței utilizatorului a jocului, care include starea de sănătate a jucătorului, inventarul și mesajele.

**AssetSetter.java** - Clasa AssetSetter stabilește resursele pentru joc.

**CollisionChecker.java** - Clasa CollisionChecker efectuează verificarea coliziunilor în joc cu patru metode: checkTile(), checkObject(), checkEntity() și checkPlayer().

**Entity.java** - Clasa Entity definește un set de variabile și metode care vor fi utilizate de alte clase precum "Player". Scopul clasei Entity este de a defini atribute și comportamente comune pentru toate entitățile jocului, cum ar fi detectarea coliziunilor, animația sprite-ului și afișarea imaginilor.

**Player.java** - Clasa Player definește obiectul Player care este controlat de jucător și este mutat în jurul lumii jocului pentru a îndeplini sarcini și a completa jocul. Obiectul Player este definit ca o subclasă a clasei Entity. Primește ca parametri de intrare obiectul GamePanel și obiectul KeyHandler.

**Tile.java** - Clasa Tile reprezintă o singură placă (tile) în joc. Are două variabile de instanță: image, un obiect BufferedImage care reprezintă imaginea afișată pentru această placă și collision, o valoare booleană care indică dacă jucătorul poate sau nu să se deplaseze pe această placă. Dacă este adevărat, înseamnă că placa este solidă și jucătorul nu poate să se deplaseze pe ea, dacă este fals, înseamnă că placa este goală și jucătorul poate să se deplaseze pe ea. Clasa nu are un constructor sau metode, astfel încât este folosită în principal ca un simplu container de date.

**TileManager.java** - Clasa TileManager gestionează plăcile (tiles) hărții jocului. Clasa conține mai multe variabile membru: gp, un obiect al clasei GamePanel, tile, un

tablou de obiecte Tile, mapTileNum, o matrice care stochează numerele de plăci ale hărții jocului.

**SuperObject.java** -reprezintă un obiect din joc și are acces la obiectul GamePanel. Clasa conține variabile membre precum image, name, collision, worldX, worldY, worldX1, worldY1, solidArea, solidAreaDefaultX și auximg. Metoda draw() desenează obiectul în grafica jocului, iar metoda drawTree() desenează un obiect de tip copac în grafica jocului. Aceste metode folosesc variabilele membre worldX, worldY, gp.tileSize și gp.player pentru a determina poziția și dimensiunea obiectului de afișat.

**Tree.java** - este o clasă care extinde clasa SuperObject și reprezintă obiectul "tree" din joc. Ea are un constructor care primește trei parametri, worldWidth, worldHeight și tileSize și setează valorile variabilelor clasei. Imaginea pentru acest obiect este încărcată dintr-un fișier folosind clasa ImageIO. Variabilele worldX și worldY sunt setate cu valori aleatorii în intervalul [0, worldWidth/tileSize) \* tileSize și [0, worldHeight/tileSize) \* tileSize. Dacă valorile generate sunt mai mici decât tileSize, atunci se adaugă diferența pentru a nu ieși din limitele hărții jocului. Variabila collision este setată la true.

**Flower.java**- reprezintă un obiect floare într-un joc și se extinde din clasa SuperObject. Clasa conține două constructori: primul constructor inițializează doar imaginea floarei și al doilea constructor inițializează și poziția floarei. Poziția este generată aleatoriu utilizând clasa Random și este setată astfel încât floarea să fie plasată pe o poziție adecvată pe hartă. Variabilele utilizate în constructor sunt worldWidth, worldHeight și tileSize, reprezentând dimensiunile lumii și dimensiunea unei plăci de joc, respectiv. Clasa are o variabilă suplimentară numită auximg, care este utilizată pentru a stoca temporar imaginea inițială a floarei înainte de a fi modificată pentru a fi afișată pe ecran. Clasa conține, de asemenea, o variabilă name care stochează numele obiectului.

**Bullet.java** -Clasa "Bullet" reprezintă o entitate de proiectil utilizată într-un joc, care extinde clasa "Projectiles". Iată o descriere a clasei: Variabila "gp" este o referință la obiectul "GamePanel" și este utilizată pentru a accesa anumite informații și funcționalități din cadrul jocului. Constructorul clasei primește ca argument un obiect "GamePanel" și inițializează variabilele specifice obiectului "Bullet". Metoda "getImage()" este responsabilă pentru încărcarea imaginii asociate proiectilului. Imaginea este extrasă dintr-un fișier și este împărțită în diferite sub-imagini pentru a fi utilizată în diferite direcții de mișcare. Clasa "Bullet" folosește metoda "ImageIO.read()" pentru a citi imaginea din fișierul "/Bullet/Bullet.png". Variabilele "up1", "up2", "down1", "down2", "right1", "right2", "left1" și "left2" reprezintă sub-imagini ale proiectilului pentru diferite direcții de mișcare. Aceste sub-imagini sunt obținute utilizând metoda "getSubimage()" pe imaginea încărcată anterior. În cazul în care apare o excepție în timpul citirii imaginii din fișier, aceasta este prinsă și afișată în consolă utilizând metoda "printStackTrace()".

**Node.java-** Clasa "Node" reprezintă un punct individual intr-un graf, folosit în clasa Pathfinder.

**PathFinder.java-** Clasa "Pathfinder" reprezintă un algoritm de căutare a drumului într-un joc și are următoarele funcționalități principale: Constructorul clasei primește un obiect "GamePanel" și inițializează variabilele asociate. Metoda "instantiateNode()" creează o matrice de obiecte "Node" bazată pe dimensiunile jocului. Metoda "resetNodes()" resetează starea nodurilor din matrice, precum și listele de noduri deschise și drumuri. Metoda "setNodes()" setează nodurile de plecare și destinație în funcție de coordonatele specificate și verifică starea nodurilor în funcție de proprietățile hărții de joc. Metoda "getCost()" calculează costurile "gCost", "hCost" și "fCost" ale unui nod pe baza poziției sale în raport cu nodul de plecare și nodul destinație. Metoda "search()" realizează algoritmul de căutare pentru a găsi cel mai scurt drum de la nodul de plecare la nodul destinație. Acesta explorează nodurile adiacente, calculează costurile și selectează cel mai bun nod pentru a continua căutarea. Metoda "openNode()" deschide un nod pentru a fi explorat în cadrul algoritmului de căutare, dacă acesta îndeplinește anumite condiții. Metoda "trackThePath()" reconstruiește drumul de la nodul destinație la nodul de plecare prin urmărirea nodurilor părinte și le adaugă în lista "pathList".

**Goblin.java-** Clasa "Goblin" este o subclasă a clasei "Entity" și reprezintă un tip de inamic numit Goblin într-un joc. Iată o descriere pe scurt a clasei: Constructorul clasei primește un obiect "GamePanel" și inițializează variabilele specifice. Metoda "getPlayerImage()" încarcă imaginile asociate sprite-ului Goblin din fișierul de resurse și le asociază variabilelor corespunzătoare pentru fiecare direcție. Metoda "update()" actualizează starea inamicului Goblin în funcție de mișcarea și acțiunile sale. Verifică distanța față de jucător și decide dacă inamicul trebuie să urmărească jucătorul sau să aibă o mișcare aleatoare. Metoda "setAction()" stabilește acțiunea inamicului Goblin în funcție de starea sa. Dacă este pe un traseu, urmărește jucătorul. Altfel, alege o direcție aleatoare sau lansează proiectile în funcție de șansele calculate aleatoriu. Clasa utilizează și un obiect "Bullet" (proiectil) pentru a gestiona lansarea de proiectile. În ansamblu, clasa "Goblin" reprezintă comportamentul și caracteristicile specifice ale inamicului Goblin în joc.

**WalkingGoblin.java**- Este la fel ca si clasa goblin, si reprezinta tipul de inamici care pot doar sa mearga dar nu sa si impuste.

**Projectiles.java -** Clasa "Projectiles" este o subclasă a clasei "Entity" și reprezintă proiectilele lansate în joc. Iată o explicație a clasei: Constructorul clasei primește un obiect "GamePanel" și inițializează variabilele specifice. Metoda "set()" este utilizată pentru a seta poziția, direcția, starea și utilizatorul proiectilului. Aceasta primește coordonatele poziției inițiale, direcția de mișcare, starea de viață a proiectilului și utilizatorul care a lansat proiectilul. Metoda "update()" actualizează starea proiectilului în fiecare cadru de joc. Verifică coliziunile cu jucătorul și cu dalele hărții și modifică poziția proiectilului în funcție de direcție. De asemenea, scade viața proiectilului și actualizează sprite-ul proiectilului. Metoda "draw()" desenează proiectilul pe ecran utilizând contextul grafic primit ca parametru. Alege imaginea sprite-ului corespunzătoare în funcție de direcția și numărul sprite-ului. Metoda "setAction()" este utilizată pentru a seta acțiunea proiectilului. Dacă proiectilul se află pe un traseu, va căuta și urmări jucătorul. În ansamblu, clasa "Projectiles" gestionează comportamentul și starea proiectilelor lansate în joc, inclusiv detectarea coliziunilor, actualizarea poziției și desenarea lor pe ecran.

**Bullet.java**- Clasa "Bullet" este o subclasă a clasei "Projectiles" și reprezintă proiectilele specifice inamice de tip "Bullet" din joc. Iată o explicație a clasei: Constructorul clasei primește un obiect "GamePanel" și inițializează variabilele specifice. Metoda "getImage()" este utilizată pentru a încărca imaginile sprite-urilor proiectilelor din fișierele sursă. Aceasta utilizează clasa "ImageIO" pentru a citi și decoda imaginile din fișiere și apoi le stochează în variabilele corespunzătoare (up1, up2, down1, down2, right1, right2, left1, left2) pentru utilizare ulterioară. Clasa "Bullet" suprascrie constructorul clasei "Projectiles" și setează valorile specifice proiectilelor de tip "Bullet", cum ar fi viteza, durata de viață, atacul și costul de utilizare. Aceasta suprascrie și variabila "name" pentru a specifica numele proiectilului ca "Bullet". Variabila "alive" este inițializată cu valoarea "false", deoarece proiectilul nu este inițial activ în joc. Clasa "Bullet" suprascrie și metoda "setAction()" pentru a specifica comportamentul proiectilului de tip "Bullet" atunci când este pe un traseu. În acest caz, nu se întâmplă nimic special în metoda "setAction()" a acestei clase. În ansamblu, clasa "Bullet" gestionează proiectilele specifice inamice de tip "Bullet" din joc, inclusiv încărcarea imaginilor, setarea valorilor specifice și comportamentul lor în cadrul jocului.

**Fountain.java-** Clasa "Fountain" este o subclasă a clasei "SuperObject" și reprezintă un obiect specific din joc numit "fountain" (fântână). Iată o explicație a clasei: Constructorul clasei inițializează variabilele specifice. Numele obiectului este setat ca "fountain". În blocul "try-catch", se încearcă încărcarea imaginii fântânii dintr-un fișier sursă utilizând clasa "ImageIO". Imaginea este citită și decodată din fișierul sursă, iar rezultatul este stocat în variabila "image" a clasei. Variabila "collision" este setată la valoarea "true", indicând că obiectul fântână are coliziune în joc. În ansamblu, clasa "Fountain" gestionează obiectul specific "fântână" din joc, inclusiv încărcarea imaginii și setarea proprietăților specifice, cum ar fi numele și coliziunea.

