**Documentatie**

1. **Enuntul problemei**

* **Tip joc**: Multiplayer (2-4 jucători) cu bile care colectează obiecte.
* **Scop**: Adună cât mai multe bile colectabile în**tr-un minut**.
* **Câștigător**: Jucătorul cu cele mai multe puncte la final.
* **Comunicare**: UDP pentru viteză.

1. **Descrierea proiectului**
2. **Implementare**

* **Server (GameServer):**
* Gestionează starea jocului (poziții jucători, scoruri, bile colectabile).
* Folosește DatagramSocket pentru comunicare.
* Trimite actualizări de stare tuturor clienților la intervale regulate.
* **Client (GameClient):**
* Afișează interfața grafică (GamePanel) și procesează input-ul utilizatorului.
* Trimite comenzi (mișcare, start joc) la server.

1. **Structuri de date**

* ConcurrentHashMap**:** Stochează jucătorii (PlayerData) și clienții (ClientInfo) pentru thread-safety.
* ArrayList<Collectible>**:** Listă de bile colectabile.
* AtomicInteger**:** Generează ID-uri unice pentru jucători și obiecte.

1. **Comunicare în rețea**

* **Protocol UDP:** Mesaje concise (e.g. "MOVE x y", "STATE P1:100,200; C1:50,60;").
* **Serializare:** Starea jocului este convertită în String pentru transmitere.

1. **Structuri de limbaj (Java) explicate**

|  |  |  |
| --- | --- | --- |
| **Structură/Concept** | **Explicație** | **Exemplu din proiect** |
| DatagramSocket | Socket UDP pentru comunicare între server și clienți. | DatagramSocket socket = new DatagramSocket(port); |
| ConcurrentHashMap | Mapă thread-safe pentru stocarea jucătorilor. | ConcurrentMap<String, PlayerData> players = new ConcurrentHashMap<>(); |
| SwingUtilities.invokeLater | Execută cod pe firul de execuție (thread) grafic (AWT/Swing). | SwingUtilities.invokeLater(() -> updateGameState(state)); |
| **Serializare mesaje** | Convertire stării jocului în String pentru transmitere. | String state = "P:" + player.id + "," + player.x + "," + player.y; |

1. **Screenshots**







