|  |  |
| --- | --- |
| Client | A videogame company |
| User | * The programmers and designers that will keep working on the project |
| Functional requirements | * R1: Register players * R3: Register loot * R4: Register enemies * R5: Increase level for a player, in case you cannot increase the level, you must inform the user what score you require to climb. * R6: Report the most repeated treasure at all levels * R7: Report the top 5 of the players according to the score. * R8: Report the treasures and enemies * R9: Report the amount of a loot found at all levels. * R10: Report the amount found of an enemy type in all levels. * R11: Classification of enemies. * R12: Report the enemy that awards the highest score and the level where it is located. * R13: Report the number of consonants found in the names of the game's enemies. |
| Problem context | A videogame company wants to develop a videogame based on levels that will change according to a score assigned to the player. |
| Non-functional requirements | * The program must use the language Java * The program must be fast, cannot longer more than 2 seconds to start * The program must work in android and web platforms |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R1: Register players | | |
| Abstract | The program must let the user set a nickname that will works as a unique identifier, a name, a score that will be initialized with 10 and lives that are going to be initialized in 5 | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| nickname | String | * If the person wants to change it. * If the nickname already exists |
| Name | String | * That the input corresponds to a String. |
| General activities needed to obtain the results | 1. The program will receive a nickname and search if it already exists. 2. The program will initialize the score and lives. | | |
| Result or post condition | The program registers a player and returns a String that will confirm the success of the operation | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| result | String | The player was successfully created. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R2: Register level | | |
| Abstract | The program must let register the different levels of the game. The level has an id and the score needed to go to the next level. Finally, the level set an amount of loot that is going to be given by the user. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| Number of enemies | int | * The type of the enemy cannot be repeated |
| Number of loots | int |  |
| General activities needed to obtain the results | 1. The program must give an identifier to the level 2. The program asks the number of enemies to the user 3. The program calculates the number of points needed to go to the next level 4. The program asks the loot in the level 5. The program must categorize all the enemies according to their level (High, medium, and low) | | |
| Result or post condition | A new level is created and returns a String confirming the success of the operation | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| Message of confirmation | String | The level was successfully created. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R3: Register loot | | |
| Abstract | The program must give loot, that are going to be bonifications to the player. Will have a name, an URL with the image of the loot, the points that is going to give to the player and random position X, Y. The levels can have more than a single treasure and this treasure can appear many times in different positions among the level. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| name | String |  |
| URL | String |
| Points given | double |
| General activities needed to obtain the results | 1. The program will receive a name, an URL and the points that the loot is going to give. 2. The program will set the loot in a random position among the screen | | |
| Result or post condition | The program creates a new loot and returns a message confirming the operation | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| Message of confirmation | String | The loot was successfully created. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R4: Register enemies | | |
| Abstract | The enemies are the entities that will reduce the score of the player. They have a name (That will work as a unique identifier), a type, the points that is going to take from the player, the points that is going to give to the player (in case the player wins) and a random position X, Y. The type of enemies in the game are ogre, magical, abstract, and boss. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| name | String |  |
| type | String |  |
| Points given | Double |  |
| Points taken | Double |  |
| General activities needed to obtain the results | 1. The program receives the name, the type, the points given, and the points taken 2. The program will generate a random position along the screen. | | |
| Result or post condition | The program creates a new enemy and returns a message confirming the operation. | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| Confirmation message | String | The enemy was successfully created. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R5: Increase level for a player, in case you cannot increase the level, you must inform the user what score you require to climb. | | |
| Abstract | The program must be able to change the level of a player when he reaches the enough points. Also, will have to inform about the points that are needed to reach another level. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
|  |  |  |
| General activities needed to obtain the results | 1. The program can change the level of a player 2. The program informs about the points needed to change level | | |
| Result or post condition | The player change level when has the enough points, the program informs to the player the levels needed | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| Points needed to change level | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R6: Report the most repeated treasure at all levels | | |
| Abstract | The program must be able to report the most repeated treasure at all levels. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
|  |  |  |
| General activities needed to obtain the results | 1. The user can select the type of report he needs from the program 2. The program returns the solution of the report needed | | |
| Result or post condition | The program returns a message with the data that is needed | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| Confirmation message | String | The data was successfully found. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R7: Report the top 5 of the players according to the score. | | |
| Abstract | The program must be able to report the 5 players with the higher score. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
|  |  |  |
| General activities needed to obtain the results | 1. Look in all the levels and return the 5 players with the highest score. | | |
| Result or post condition | The program returns a message with the 5 best players in the game | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| Message with the 5 best players | String | * The players cannot be repeated. |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R8: Report loots and enemies | | |
| Abstract | The program must be able to report loots and enemies (separated by comma) of a level given by the user | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| Level name | String | * The entity exists. |
| General activities needed to obtain the results | * The program will look for a level that has the same name that was given. | | |
| Result or post condition | The program returns a message with the loots and enemies (separated by comma) from the level given by the user. | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| Confirmation message | String | * The level was successfully found |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R9: Report the amount of a loot found at all levels. | | |
| Abstract | The program must be able to return all the loot specified by the user in all levels. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| Name of the loot | String | * The entity exists. |
| General activities needed to obtain the results | * The program will look for an object that has the same name that was given. | | |
| Result or post condition | The program returns a message with the object needed. | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| Confirmation message | String | * The object was successfully found |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R10: Report the amount found of an enemy type in all levels | | |
| Abstract | The program must be able to report the amount found of an enemy type in all levels, that is, if the user wants to know how many ogres exist in all levels. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
| Enemy type | String | The enemy exists. |
| General activities needed to obtain the results | * The program will look for an enemy that has the same name that was given. | | |
| Result or post condition | The program returns a message with the number of the enemy type needed. | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| Confirmation message | String | * The object was successfully found |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R11: Classification of enemies | | |
| Abstract | The program must be able to give a class to each enemy corresponding to the score they give and take. (High, medium, and low) if the points that the loot drops are higher than the points the enemy takes, its level will be low, if they are equal the level will be medium and if the loot is lower than the points it gives, the level will be High. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
|  |  |  |
| General activities needed to obtain the results | * The program classifies each enemy corresponding to the points that it takes, or it gives. | | |
| Result or post condition | The program classifies each enemy in High, medium or low. | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R12: Report the enemy that awards the highest score and the level where it is located. | | |
| Abstract | The program must return the enemy with the highest score and the level where it is located. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
|  |  |  |
| General activities needed to obtain the results | * The searches in all levels for the enemy with the highest score and returns it with the level it is located. | | |
| Result or post condition | The enemy with the highest score with its level is found. | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| The enemy with the highest score | Enemy | The enemy exists and is in a level |
| The level where the enemy is located | Level |

|  |  |  |  |
| --- | --- | --- | --- |
| Name or identifier | R13: Report the number of consonants found in the names of the game's enemies. | | |
| Abstract | The program must return the consonants found in all the enemies name of the game. | | |
| Inputs | **Input name** | **Data type** | **Condition of select or repetition** |
|  |  |  |
| General activities needed to obtain the results | * The searches in all levels for the consonants of the enemies’ names. | | |
| Result or post condition | The count of the consonants of the enemies’ names in all levels. | | |
| Outputs | **Output name** | **Data type** | **Condition of select or repetition** |
| The count of enemies’ names | String | The enemy’s name cannot be null |

|  |  |  |
| --- | --- | --- |
| **Functional requirements** | **Class name** | **Method name** |
| R1: Register Player | Player class | Player(…) |
| Level class | addPlayer(…) |
| Game class | addPLayerToLevel(Level) |
| Main | registerPlayer(Level) |
| R2: Register Level | Level class | Level(…) |
| Game class | addLevel(…) |
| Main | registerLevel(…) |
| R3: Register Loot | Loot class | Loot(…) |
| Level class | addLoot(…) |
| Game class | addLootToLevel(…) |
| Main | registerLootInLevel(…) |
| R4: Register Enemies | Enemy class | Enemy(…) |
| Level class | addEnemy(…) |
| Game class | addEnemyToLevel(…) |
| Main | registerEnemy(…) |
| R5: Increase level for a player, in case you cannot increase the level, you must inform the user what score you require to climb. | Player class | setPlayerScore(…) |
| Level class | getPlayerByNickname(nickname : String) |
| Game class | increasePlayerLevel(levelID) |
| Main | increasePlayerLevel() |
| R6: Report the most repeated treasure at all levels | Loot class | getLootName(…) |
| Level class | countLootByName(…) |
| Game class | countAllLootsInLevels(…) |
| Main | countAllLootsInLevels(…) |
| R7: Report the top 5 of the players according to the score. | Player class | getPlayerScore(…) |
| Level class | maxScorePlayer() |
| Game class | maxScorePlayerAllLevels(…) |
| Main | getTopFivePlayers() |
| R8: Report the treasures and enemies | Loot class | getLootName() |
| Enemy class | getEnemyName() |
| Level class | getAllEnemies() |
| Level class | getAllLoots() |
| Game class | getAllEnemiesAndLoots(Level) |
| Main | reportAllEnemiesAndLoots(Level) |
| R9: Report the amount of a loot found at all levels. | Loot class | getLootName() |
| Level class | countLootByName(lootName) |
| Game class | getAllLootsInLevel(lootName) |
| Main | amountOfLootInLevel(lootName) |
| R10: Report the amount found of an enemy type in all levels. | Enemy class | getEnemyType() |
| Level class | countEnemyByType() |
| Game class | countEnemyTypeAllLevels() |
| Main | countEnemyTypeAllLevels() |
| R11: Classification of enemies. | Enemy class | getPointsTaken() |
|  | Enemy class | getPointsGiven() |
|  | Enemy class | setDifficultLevel() |
| R12: Report the enemy that awards the highest score and the level where it is located. | Enemy class | getEnemyDifficultLevel() |
| Enemy class | getPointsTaken() |
| Level class | getMaxEnemyScore() |
|  | Game class | getMaxEnemyScoreAllLevels() |
|  | Main | getMaxEnemyScoreAllLevels() |
| R13: Report the number of consonants found in the names of the game's enemies. | Enemy class | getEnemyName() |
|  | Level class | getEnmiesName() |
|  | Game class | countConsonantsAllEnemiesNames() |
|  | Main | countConsonantsInAllEnemiesNames() |