MiniProject level 7

December 6, 2022

1 Miniproject Assessed Exercise

2 Level 7

2.1 Asad Ali Khan

2.2 Student ID: 220257466

2.3 06 December 2022

2.4 Version 1

2.5 Summary of the Question

Write a game that allows you to explore a castle, (or similar), collecting objects and disabling traps.

2.6 Justification that the program passes this level

This is based on tick boxes at start of the question description for miniproject program level 1 - I can explain how the program works - I wrote this program myself - My program compiles and runs in JHUB - My program has screen output and keyboard input - in createNewPlayer - My program has variables, assignment and expressions - in startNewGame, createNewSword, createNewShield etc.. - My program has at least one method that returns a result - in getRandomNum, itemDropped etc.. - My program has decision statements - in askUserToEquip - My program has methods that take argument(s) - in displayShield, displaySword etc.. - My program includes Loops - in startNewGame - My program has final variables - in getRandomNum - My program includes arrays, accessed with a loop - in readSaveFile - My program includes Loops within loops. - in fightSequence - My program uses methods including one that is passed and uses array arguments - in loadSword, loadShield - My program includes records with no methods and just field definitions - Player, Inventory, Enemy, Sword, Shield, Food, Key, Room, GameInfo - My program has both file input AND file output - saveGame, readSaveFile - My program includes methods doing a well-defined task and uses a sequence of method calls - seen below - My program include simple comments including my name, student number and date - I have written a literate version of the code - documenting each method - including a test plan for each method - included a test plan for the whole program - included a full version of the program at the end

2.7 The literate program development

2.7.1 Records

What it does These are all records defined as a special class with no methods just field definitions. #### Implementation (how it works) Each one stores information about specific objects, entities and general game information during the program is running

[3]:
```
class Enemy {

    String name;
    int health;
    int damage;
    int defence;
}

class Sword {

    String name;
    int durability;
    int damage;
}

class Shield {

    String name;
    int durability;
    int defence;
}

class Food {

    String name;
    int healthRegen;
}

class Key {

    String name;
    int disarmChance;
}

class Room {

    int enemies;
    boolean isTrap;
}
```

```
class Inventory {

    Sword sword;
    Shield shield;
    Key key;
}

class Player {

    String name;
    int health;
    int damage;
    int defence;
    Inventory inventory = new Inventory();
}

class GameInfo {

    int maximumHealth; // minium health of spawned enemies
    int maximumDamage; // minimum damage of spawned enemies
    int maximumDefence; // maximum defence of spawned enemies
    int rooms; // stores the number of rooms that the player successfully passed
    boolean gameover; // stores wether the game is over or not
}
```

2.7.2 printString

What it does This method prints the string that is passed into it (Makes printing strings easier as you do not need to type System.out.println every time) #### Implementation (how it works) simple print statement

[6]:
```
public static void printString(String string) {
        // prints the string that is passed into it
        System.out.println(string);
    }
```

Testing

[78]:
```
printString("Welcome to booby-trap castle");
```

```
Welcome to booby-trap castle
```

2.7.3 pressEnterToContinue

What it does Asks the user to press enter when they would like to continue #### Implementation (how it works) Uses the scanner class to take an input and they uses a try and exception block to when reading a new line when taking the input and avoids the error

[7]:
```
public static void pressEnterToContinue() {
        // waits for the user to enter a new line to continue to the next action
        printString("Press Enter To Continue");
        Scanner scanner = new Scanner(System.in);
        try {
            System.in.read();
            scanner.nextLine();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Testing

[80]:
```
pressEnterToContinue();
```

```
Press Enter To Continue
```

2.7.4 clearScreen

What it does This method clears the terminal so that it is easier to read #### Implementation (how it works) prints a specific string that flushes the terminal

[13]:
```
public static void clearScreen() {
        // flushes the screen of its text to make the game easier to read
        printString("\033[H\033[2J");
    }
```

Testing

[14]:
```
clearScreen();
```

2.7.5 createNewGameInfo

What it does Creates a new GameInfo variable which stores information about the game #### Implementation (how it works) Takes the values that are passed into it and stores it in its relevant fields

[15]:
```
public static GameInfo createNewGameInfo(int maximumHealth,
                                          int maximumDamage,
                                          int maximumDefence,
                                          int rooms) {
        // creates a new gameInfo variable
        GameInfo gameInfo = new GameInfo();
        gameInfo.gameover = false;

        gameInfo.rooms = rooms;
        gameInfo.maximumHealth = maximumHealth;
        gameInfo.maximumDamage = maximumDamage;
        gameInfo.maximumDefence = maximumDefence;

        return gameInfo;
    }
```

Testing

[16]:
```
createNewGameInfo(10, 2, 1, 0);
```

[16]:
```
REPL.$JShell$20$GameInfo@284391ea
```

2.7.6 createNewPlayer

What it does Creates a new Player variable which stores information about the player #### Implementation (how it works) Takes the values that are passed into it and stores it in its relevant fields

[17]:
```
public static Player createNewPlayer(String name, int health, int damage, int␣
↪defence) {
        // Creates a new player
        Player player = new Player();
        player.name = name;
        player.health = health;
        player.damage = damage;
        player.defence = defence;

        return player;
    }
```

Testing

[18]:
```
createNewPlayer("Asad", 15, 3, 1);
```

[18]:
```
REPL.$JShell$12$Player@2e5c3696
```

2.7.7 getRandomNum

What it does Generates a random number #### Implementation (how it works) Takes a number to determine the number of sides a “dice” has which is then “rolled” and the number is then returned

[19]:
```
public static int getRandomNum(int num) {
        // rolls a dice with the number of sides passed to it
        Random random = new Random();
        final int rngNum = random.nextInt(num);

        return rngNum;
    }
```

Testing

[20]:
```
getRandomNum(7);
```

[20]:
```
4
```

2.7.8 createNewSword

What it does Creates a new Sword variable which stores information about the dropped sword #### Implementation (how it works) Takes the gameInfo variable and generates stats randomly based on the gameInfo variable and stores it in its relevant fields

[25]:
```
public static Sword createNewSword(GameInfo gameInfo) {
        // this creates a new sword
        Sword sword = new Sword();

        String[] names = {
                "Long Sword",
                "Dagger",
                "Short Sword",
                "Mace",
                "Katana",
                "Battle Axe",
        };

        sword.name = names[getRandomNum(names.length)];
        sword.durability = getRandomNum(7) + 1;
        sword.damage = getRandomNum((int) (gameInfo.maximumHealth / 5.0)) + 1;

        return sword;
    }
```

Testing

[26]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
createNewSword(gameInfo)
```

[26]:
```
REPL.$JShell$14$Sword@22d89dcc
```

2.7.9 createNewShield

What it does Creates a new Shield variable which stores information about the dropped shield #### Implementation (how it works) Takes the gameInfo variable and generates stats randomly based on the gameInfo variable and stores it in its relevant fields

[28]:
```
public static Shield createNewShield(GameInfo gameInfo) {
        // this creates a new shield
        Shield shield = new Shield();

        String[] names = {
                "Broad Shield",
                "Enchanted Shield",
                "Steel Plated Shield",
        };

        shield.name = names[getRandomNum(names.length)];
        shield.durability = getRandomNum((int) (3)) + 1;
        shield.defence = getRandomNum((int) (gameInfo.maximumHealth / 5.0)) + 1;

        return shield;
    }
```

Testing

[29]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
createNewShield(gameInfo)
```

[29]:
```
REPL.$JShell$15$Shield@6505bf21
```

2.7.10 createNewFood

What it does Creates a new Food variable which stores information about the dropped food item #### Implementation (how it works) Takes the gameInfo variable and generates stats randomly based on the gameInfo variable and stores it in its relevant fields

[32]:
```
public static Food createNewFood(GameInfo gameInfo) {
        // this creates a new food item
        Food food = new Food();

        String[] names = { "Beef Jerky", "Apple pie", "Hearty bread" };

        food.name = names[getRandomNum(names.length)];
        food.healthRegen = getRandomNum((int) (gameInfo.maximumHealth / 3.5)) +␣
 ↪1;

        return food;
    }
```

Testing

[33]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
createNewFood(gameInfo)
```

[33]:
```
REPL.$JShell$16$Food@1993b37
```

2.7.11 createNewKey

What it does Creates a new Key variable which stores information about the dropped key #### Implementation (how it works) Generates stats randomly and stores it in its relevant fields. The chance is a random numnber between 1 - 100.

[82]:
```
public static Key createNewKey() {
        // this creates a new key
        Key key = new Key();

        String[] names = { "Gold key", "Old key", "Enchanted key" };

        key.name = names[getRandomNum(names.length)];
        key.disarmChance = getRandomNum(100) + 1;

        return key;
    }
```

Testing

[83]:
```
createNewKey();
```

[83]:
```
REPL.$JShell$17$Key@45da4feb
```

2.7.12 createNewEnemy

What it does Creates a new Enemy variable which stores information about the enemy #### Implementation (how it works) Takes the gameInfo variable and generates stats randomly based on the gameInfo variable and stores it in its relevant fields

[115]:
```
public static Enemy createNewEnemy(GameInfo gameInfo) {
        // this creates a new enemy
        Enemy enemy = new Enemy();

        String[] names = {
                "Slime",
                "Zombie",
                "Skeleton",
                "Ogre",
                "Troll",
                "Goblin",
                "Witch",
        };

        enemy.name = names[getRandomNum(names.length)];
        enemy.health = getRandomNum(gameInfo.maximumHealth) + 1;
        enemy.damage = getRandomNum(gameInfo.maximumDamage) + 1;
        enemy.defence = getRandomNum(gameInfo.maximumDefence);

        return enemy;
    }
```

Testing

[150]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
createNewEnemy(gameInfo);
```

[150]:
```
REPL.$JShell$13$Enemy@d6a1540
```

2.7.13 determineIfTrappedRoom

What it does Determine whether a room will have traps or not #### Implementation (how it works) calls the getRandomNum method with a chance of 1/5 that a room has traps

[124]:
```
public static Boolean determineIfTrappedRoom() {
        // this rolls a 5 sided dice, if the rolled number is 0 then the next␣
 ↪room is a
        // trap
        int numberRolled = getRandomNum(5);

        if (numberRolled == 0) {
            return true;
        } else {
            return false;
        }
    }
```

Testing

[125]:
```
determineIfTrappedRoom();
```

[125]:
```
false
```

2.7.14 determineIfKeyWorks

What it does Determine whether a key will be successful or not #### Implementation (how it works) calls the getRandomNum method to return a number between 1-100. This number is then compared with the key's disarmChance - if < than the disarmChance then the key works.

[137]:
```
public static Boolean determineIfKeyWorks(Key key) {
        // compares the success rate with a random number between 0-100, if its␣
 ↪more
        // then the key works
        int numberRolled = getRandomNum(99) + 1;

        if (numberRolled <= key.disarmChance) {
            return true;
        } else {
            return false;
        }
    }
```

Testing

[138]:
```
Key key = createNewKey();
determineIfKeyWorks(key);
```

[138]:
```
false
```

2.7.15 createNewRoom

What it does Creates a new Room variable which stores information about the rooom #### Implementation (how it works) Generates stats randomly and stores it in its relevant fields. The chance is a random numnber between 1 - 2. Also determines whether its a trapped room or not.

[126]:
```
public static Room createNewRoom() {
        // this creates a new room with the number of enemies specified
        Room room = new Room();
        room.enemies = getRandomNum(2) + 1;
        room.isTrap = determineIfTrappedRoom();
        return room;
    }
```

Testing

[127]:
```
createNewRoom();
```

[127]:
```
REPL.$JShell$18$Room@3557b4e7
```

2.7.16 itemDropped

What it does Determines whether an item is dropped when the user enters a new room #### Implementation (how it works) calls the getRandomNum method with a chance of 1/5 that an item is dropped. Returns true or false based on the number that is rolled

[34]:
```
public static boolean itemDropped() {
        // rolls a 5 sided dice to determine if an item was dropped
        int numberRolled = getRandomNum(5);

        if (numberRolled == 0) {
            return true;
        } else {
            return false;
        }
    }
```

Testing

[35]:
```
itemDropped();
```

[35]:
```
false
```

2.7.17 getRandomDroppedItemType

What it does Determines what type of item is dropped when an item is dropped #### Implementation (how it works) calls the getRandomNum method with a chance of 1/4 that an item of each item is dropped. Returns as the string value of the item type

[36]:
```
public static String getRandomDroppedItemType() {
        // rolls a 4 sided dice to determine what type of item was dropped
        int numberRolled = getRandomNum(4);

        if (numberRolled == 0) {
            return "sword";
        } else if (numberRolled == 1) {
            return "shield";
        } else if (numberRolled == 2) {
            return "food";
        } else {
            return "key";
        }
    }
```

Testing

[37]:
```
getRandomDroppedItemType();
```

[37]:
```
key
```

2.7.18 getRandomLostItemType

What it does Determines what type of item is lost during a trap #### Implementation (how it works) calls the getRandomNum method with a chance of 1/2 either a sword or shield is lost. Returns a string of the item

[169]:
```
public static String getRandomLostItemType() {
        // rolls a 2 sided dice to determine what type of item was dropped
        int numberRolled = getRandomNum(2);

        if (numberRolled == 0) {
            return "sword";
        } else {
            return "shield";
        }
    }
```

Testing

[171]:
```
getRandomLostItemType();
```

[171]:
```
sword
```

2.7.19 displayDroppedSword

What it does Displays the stats of a dropped sword #### Implementation (how it works) Uses print statements to display information about the specific sword that is passed into it

[41]:
```
public static void displayDroppedSword(Sword sword) {
        // displays the sword that is dropped
        printString("-----------------");
        printString("A Sword Dropped!");
        printString(sword.name + " (+ " + sword.damage + ") " + sword.
  ↪durability);
        printString("-----------------");
    }
```

Testing

[42]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Sword droppedSword = createNewSword(gameInfo);
displayDroppedSword(droppedSword);
```

```
-----------------
A Sword Dropped!
Mace (+ 2) 4
-----------------
```

2.7.20 displayDroppedShield

What it does Displays the stats of a dropped shield #### Implementation (how it works) Uses print statements to display information about the specific shield that is passed into it

[45]:
```
public static void displayDroppedShield(Shield shield) {
        // displays the shield that is dropped
        printString("-----------------");
        printString("A Shield Dropped!");
        printString(shield.name + " (+ " + shield.defence + ") " + shield.
↪durability);
        printString("-----------------");
    }
```

Testing

[46]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Shield droppedShield = createNewShield(gameInfo);
displayDroppedShield(droppedShield);
```

```
-----------------
A Shield Dropped!
Steel Plated Shield (+ 2) 2
-----------------
```

2.7.21 displayDroppedFood

What it does Displays the stats of a dropped food item #### Implementation (how it works) Uses print statements to display information about the specific food that is passed into it

[47]:
```
public static void displayDroppedFood(Food food) {
        // displays the food that is dropped
        printString("-----------------");
        printString("Some food Dropped!");
        printString(
                "You ate the " +
                        food.name +
                        " and healed " +
                        food.healthRegen +
                        " points.");
        printString("-----------------");
    }
```

Testing

[49]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Food droppedFood = createNewFood(gameInfo);
displayDroppedFood(droppedFood);
```

```
-----------------
Some food Dropped!
You ate the Apple pie and healed 2 points.
-----------------
```

2.7.22 displayDroppedKey

What it does Displays the stats of a dropped key #### Implementation (how it works) Uses print statements to display information about the specific key that is passed into it

[155]:
```
public static void displayDroppedKey(Key key) {
        // displays the key that is dropped
        printString("----------------");
        printString("A Key Dropped!");
        printString(key.name);
        printString("Success Chance: " + key.disarmChance + "%");
        printString("----------------");
    }
```

Testing

[156]:
```
Key droppedKey = createNewKey();
displayDroppedKey(droppedKey);
```

```
----------------
A Key Dropped!
Gold key
Success Chance: 57%
----------------
```

2.7.23 displayPlayerStats

What it does Displays the stats of the player #### Implementation (how it works) Uses print statements to display information about the player

[51]:
```
public static void displayPlayerStats(Player player) {
        // displays the players stats
        printString("----------------");
        printString("name: " + player.name);
        printString("health: " + player.health);
        printString("damage: " + player.damage);
        printString("defence: " + player.defence);
        printString("----------------");
    }
```

Testing

[52]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
displayPlayerStats(player)
```

```
----------------
name: Asad
health: 15
damage: 3
```

```
defence: 1
-----------------
```

2.7.24 displaySword

What it does Displays the stats of an equipped sword #### Implementation (how it works) Uses print statements to display information about the equipped sword - unless there is no sword equipped. If this is the case then it just returns "sword: empty"

[53]:
```
public static void displaySword(Player player) {
        // displays an equipped sword
        if (player.inventory.sword == null) {
            printString("sword: empty");
        } else {
            printString("sword: "
                        + player.inventory.sword.name
                        + " (+ "
                        + player.inventory.sword.damage
                        + ") "
                        + player.inventory.sword.durability);
        }
    }
```

Testing

[56]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Player player = createNewPlayer("Asad", 15, 3, 1);
displaySword(player);
Sword droppedSword = createNewSword(gameInfo);
player.inventory.sword = droppedSword;
displaySword(player);
```

```
sword: empty
sword: Long Sword (+ 2) 3
```

2.7.25 displayShield

What it does Displays the stats of an equipped shield #### Implementation (how it works) Uses print statements to display information about the equipped shield - unless there is no sword equipped. If this is the case then it just returns "shield: empty"

[68]:
```
public static void displayShield(Player player) {
        // displays an equipped shield
        if (player.inventory.shield == null) {
            printString("shield: empty");
        } else {
            printString("shield: "
                        + player.inventory.shield.name
                        + " (+ "
                        + player.inventory.shield.defence
                        + ") "
                        + player.inventory.shield.durability);
        }
    }
```

Testing

[69]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Player player = createNewPlayer("Asad", 15, 3, 1);
displayShield(player);
Shield droppedShield = createNewShield(gameInfo);
player.inventory.shield = droppedShield;
displayShield(player);
```

```
shield: empty
shield: Enchanted Shield (+ 1) 3
```

2.7.26 displayKey

What it does Displays the stats of an equipped key #### Implementation (how it works) Uses print statements to display information about the equipped key - unless there is no key equipped. If this is the case then it just returns "key: empty"

[72]:
```
public static void displayKey(Player player) {
        // displays an equipped shield
        if (player.inventory.key == null) {
            printString("key: empty");
        } else {
            printString("key: "
                        + player.inventory.key.name
                        + " ("
                        + player.inventory.key.disarmChance
                        + "%)");
        }
    }
```

Testing

[91]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Player player = createNewPlayer("Asad", 15, 3, 1);
displayKey(player);
Key droppedKey = createNewKey();
player.inventory.key = droppedKey;
displayKey(player);
```

```
key: empty
key: Enchanted key (97%)
```

2.7.27 displayPlayerInventory

What it does Displays the stats of the player's inventory #### Implementation (how it works) Uses print statements to display information about the player's inventory as well as function calls.

[93]:
```
public static void displayPlayerInventory(Player player) {
        // displays the players inventory
        printString("-----------------");
        printString("    Inventory    ");
        displaySword(player);
        displayShield(player);
        displayKey(player);
        printString("-----------------");
    }
```

Testing

[103]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(50, 20, 10, 0);
displayPlayerInventory(player);
Key droppedKey = createNewKey();
player.inventory.key = droppedKey;
Shield droppedShield = createNewShield(gameInfo);
player.inventory.shield = droppedShield;
Sword droppedSword = createNewSword(gameInfo);
player.inventory.sword = droppedSword;
displayPlayerInventory(player);
```

```
-----------------
    Inventory
sword: empty
shield: empty
key: empty
-----------------
-----------------
    Inventory
sword: Battle Axe (+ 6) 2
shield: Steel Plated Shield (+ 9) 3
key: Gold key (55%)
-----------------
```

2.7.28 displayEnemy

What it does Displays an enemies stats #### Implementation (how it works) Uses print statements to display information about the enemy

[151]:
```
public static void displayEnemy(Enemy enemy) {
        // displays an enemys stats
        printString("-----------------");
        printString("name: " + enemy.name);
        printString("health: " + enemy.health);
        printString("damage: " + enemy.damage);
        printString("defence: " + enemy.defence);
        printString("-----------------");
    }
```

Testing

[152]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Enemy enemy = createNewEnemy(gameInfo);
displayEnemy(enemy);
```

```
-----------------
name: Goblin
health: 1
damage: 2
defence: 0
-----------------
```

2.7.29 displayScore

What it does Displays the score #### Implementation (how it works) Uses print statements to display the score after reading the number of rooms

[157]:
```
public static void displayScore(GameInfo gameInfo) {
        // displays the current room/score
        printString("-----------------");
        printString("Score: " + gameInfo.rooms);
        printString("-----------------");
    }
```

Testing

[158]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 7);
displayScore(gameInfo);
```

```
-----------------
Score: 7
-----------------
```

2.7.30 displayMainMenu

What it does Displays the main menu #### Implementation (how it works) Uses print statements to displays the possible options and the correspond button to press to select the option

[159]:
```
public static void displayMainMenu() {
        // displays the Main menu
        printString("-----------------");
        printString("Menu");
        printString("(1) Start new game");
        printString("(2) Load game");
        printString("-----------------");
    }
```

Testing

[160]:
```
displayMainMenu();
```

```
-----------------
Menu
(1) Start new game
(2) Load game
-----------------
```

2.7.31 displayHud

What it does Displays the HUD to the user #### Implementation (how it works) Calls the displayScore function and displayPlayerInventory to display the HUD

[162]:
```
public static void displayHud(GameInfo gameInfo, Player player) {
        // displays the score and inventory together
        displayScore(gameInfo);
        displayPlayerInventory(player);
    }
```

Testing

[163]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(50, 20, 10, 0);
displayHud(gameInfo, player);
```

```
-----------------
Score: 0
-----------------
-----------------
    Inventory
sword: empty
shield: empty
key: empty
-----------------
```

2.7.32 askMenuOption

What it does Asks the user for their option #### Implementation (how it works) Takes a user input and uses a while loop to make sure the input is valid. Returns true if its 1, false if else

[164]:
```java
public static boolean askMenuOption() {
        // asks the user what option they would like to proceed with
        Scanner userinput = new Scanner(System.in);
        printString("Select your option");
        String answer = userinput.nextLine();

        while (!answer.equals("1") & !answer.equals("2")) {
            printString("Invalid input(1/2)");
            printString("Select your option");
            answer = userinput.nextLine();
        }

        if (answer.equals("1")) {
            return true;
        } else {
            return false;
        }
    }
```

Testing

[165]:
```java
askMenuOption();
```

```
Select your option

 3

Invalid input(1/2)
Select your option

 a

Invalid input(1/2)
Select your option

 1
```

[165]:
```
true
```

[166]:
```java
askMenuOption();
```

```
Select your option

 2
```

[166]:
```
false
```

2.7.33 askUserToEquip

What it does Asks the user whether they want to equip the dropped item #### Implementation (how it works) takes a user input, uses a while loop to check whether the input is valid or not (not “yes” or “no”). It then returns true or false based on the user input

[104]:
```
public static boolean askUserToEquip() {
        // asks the user whether they would like to equip the dropped item
        Scanner userinput = new Scanner(System.in);
        printString("Would you like to equip the item");
        String answer = userinput.nextLine();

        while (!answer.equals("yes") & !answer.equals("no")) {
            printString("Invalid input(yes/no)");
            printString("Would you like to equip the item");
            answer = userinput.nextLine();
        }

        if (answer.equals("yes")) {
            return true;
        } else {
            return false;
        }
    }
```

Testing

[107]:
```
askUserToEquip();
```

```
Would you like to equip the item

 a

Invalid input(yes/no)
Would you like to equip the item

 1

Invalid input(yes/no)
Would you like to equip the item

 yes
```

[107]:
```
true
```

[108]:
```
askUserToEquip();
```

```
Would you like to equip the item

 no
```

[108]:
```
false
```

2.7.34 eatFood

What it does Eats the dropped food #### Implementation (how it works) Adds the food's health regen value to the total player health

[109]:
```
public static void eatFood(Player player, Food food) {
        // players health increases with the food
        player.health += food.healthRegen;
    }
```

Testing

[111]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Player player = createNewPlayer("Asad", 15, 3, 1);
displayPlayerStats(player);
Food droppedFood = createNewFood(gameInfo);
displayDroppedFood(droppedFood);
eatFood(player, droppedFood);
displayPlayerStats(player);
```

```
-----------------
name: Asad
health: 15
damage: 3
defence: 1
-----------------
-----------------
Some food Dropped!
You ate the Beef Jerky and healed 2 points.
-----------------
-----------------
name: Asad
health: 17
damage: 3
defence: 1
-----------------
```

2.7.35 askPlayerName

What it does Asks the user for their name #### Implementation (how it works) Uses the scanner class to take a user input and returns the result

[139]:
```
public static String askPlayerName() {
        // takes a user input for the players name and returns it
        String name;
        Scanner userinput = new Scanner(System.in);

        printString("What is your name: ");
        name = userinput.nextLine();
        return name;

    }
```

Testing

[140]:
```
askPlayerName();
```

```
What is your name:
```

```
 Asad
```

[140]:
```
Asad
```

2.7.36 askFileName

What it does Asks the user for a file and returns the file name #### Implementation (how it works) Uses the scanner class to take an input and returns the result.

[141]:
```
public static String askFileName() {
        // asks the suer to enter a file and returns the filename
        String fileName;
        Scanner userinput = new Scanner(System.in);

        printString("Save file name: ");
        fileName = userinput.nextLine();

        return fileName;

    }
```

Testing

[142]:
```
askFileName();
```

```
Save file name:
```

```
 game1
```

[142]:
```
game1
```

2.7.37 increaseBaseStats

What it does Takes a player variable and increases its stats based on a dice roll #### Implementation (how it works) calls the getRandomNum method with a chance of 1/2 of increasing the defence stat or damage stat.

[144]:
```
public static void increaseBaseStats(Player player) {
        // rolls a 2 sided dice to determine what type of item was dropped
        int numberRolled = getRandomNum(2);
        printString("-----------------");
        if (numberRolled == 0) {
            printString("Your damage increased");
            player.damage += 1;
        } else {
            printString("Your defence increased");
            player.defence += 1;
        }
        printString("-----------------");
    }
```

Testing

[145]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
displayPlayerStats(player);
increaseBaseStats(player);
displayPlayerStats(player);
```

```
-----------------
name: Asad
health: 15
damage: 3
defence: 1
-----------------
-----------------
Your damage increased
-----------------
-----------------
name: Asad
health: 15
damage: 4
defence: 1
-----------------
```

2.7.38 increaseEnemyDifficulty

What it does Takes an enemy and increases all its stats randomly #### Implementation (how it works) Calls the getRandomNum function to increase its stats by a range of 2 -(2 + 20% of the enemies stat)

[147]:
```
public static void increaseEnemyDifficulty(Enemy enemy) {
        // takes the enemy and increases its difficulty
        printString(enemy.name + " gets stronger!");
        enemy.health += getRandomNum((enemy.health / 5) + 2);
        enemy.damage += getRandomNum((enemy.damage / 5) + 2);
        enemy.defence += getRandomNum((enemy.defence / 5) + 2);
    }
```

Testing

[153]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Enemy enemy = createNewEnemy(gameInfo);
displayEnemy(enemy);
increaseEnemyDifficulty(enemy);
displayEnemy(enemy);
```

```
-----------------
name: Troll
health: 3
damage: 2
defence: 0
-----------------
Troll gets stronger!
-----------------
name: Troll
health: 3
damage: 3
defence: 1
-----------------
```

2.7.39 increaseGameDifficulty

What it does Takes a player GameInfo and increases its stats based on a dice roll #### Implementation (how it works) Calls the getRandomNum method with a chance of 1/2 of increasing the maxDefence stat or maxDamage stat. Also increments the room and maxHealth

[167]:
```
public static void increaseGameDifficulty(GameInfo gameInfo) {
        // takes the gameInfo and increases its difficulty and increments the␣
 ↪room
        gameInfo.rooms += 1;
        gameInfo.maximumHealth += 2;
        if (getRandomNum(2) == 0) {
            gameInfo.maximumDamage += 1;
        } else {
            gameInfo.maximumDefence += 1;
        }

    }
```

Testing

[168]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
increaseGameDifficulty(gameInfo);
```

2.7.40 loadSword

What it does Creates a sword with the data given #### Implementation (how it works) Takes the values and assigns the values to its corresponding field

[181]:
```
public static Sword loadSword(String name, int[] dataArray) {
    // loads a sword based on the data passed into it
        Sword sword = new Sword();
        sword.name = name;
        sword.durability = dataArray[0];
        sword.damage = dataArray[1];

        return sword;

    }
```

Testing

[182]:
```
int[] dataArray = {3, 3};
loadSword("Sword", dataArray);
```

[182]:
```
REPL.$JShell$14$Sword@5ae099c9
```

2.7.41 loadShield

What it does Creates a shield with the data given #### Implementation (how it works) Takes the values and assigns the values to its corresponding field

[183]:
```
public static Shield loadShield(String name, int[] dataArray) {
    // loads a shield based on the data passed into it
        Shield shield = new Shield();
        shield.name = name;
        shield.durability = dataArray[0];
        shield.defence = dataArray[1];

        return shield;

    }
```

Testing

[184]:
```
int[] dataArray = {3, 3};
loadShield("Shield", dataArray);
```

[184]:
```
REPL.$JShell$15$Shield@352305d1
```

2.7.42 loadKey

What it does Creates a key with the data given #### Implementation (how it works) Takes the values and assigns the values to its corresponding field

[185]:
```
public static Key loadKey(String name, int disarmChance) {
    // loads a key based on the data passed into it
        Key key = new Key();
        key.name = name;
        key.disarmChance = disarmChance;

        return key;

    }
```

Testing

[186]:
```
loadKey("Key", 42)
```

[186]:
```
REPL.$JShell$17$Key@241e389a
```

2.7.43 gameOverMessage

What it does Displays a game over message #### Implementation (how it works) Takes a prompt to display the reason of game over using print functions

[187]:
```
public static void gameOverMessage(String prompt) {
        // prints out a gameover message
        printString("-=-=-=-=-=-=-=-=-=-");
        printString(prompt);
        printString("-=-=-=-=-=-=-=-=-=-");
    }
```

Testing

[188]:
```
gameOverMessage("Died to a trap")
```

```
-=-=-=-=-=-=-=-=-=-
Died to a trap
-=-=-=-=-=-=-=-=-=-
```

2.7.44 droppedItemSequence

What it does Carries out the dropped item sequence, where it asks the user whether they would like to equip the item that is dropped #### Implementation (how it works) Uses an if statement for each of the item types. The dropped item type is determined from the function getRandomDroppedItemType. It then creates a item and asks if they would like to equip.

[192]:
```
public static void droppedItemSequence(Player player, GameInfo gameInfo) {
        // asks the user whether they would like to equip the dropped item
        String droppedItemType = getRandomDroppedItemType();

        if (droppedItemType.equals("sword")) {
            Sword droppedSword = createNewSword(gameInfo);
            displayDroppedSword(droppedSword);
            if (askUserToEquip()) {
                player.inventory.sword = droppedSword;
            }
        } else if (droppedItemType.equals("shield")) {
            Shield droppedShield = createNewShield(gameInfo);
            displayDroppedShield(droppedShield);
            if (askUserToEquip()) {
                player.inventory.shield = droppedShield;
            }
        } else if (droppedItemType.equals("food")) {
            Food droppedFood = createNewFood(gameInfo);
            displayDroppedFood(droppedFood);
            eatFood(player, droppedFood);
        } else if (droppedItemType.equals("key")) {
            Key droppedKey = createNewKey();
            displayDroppedKey(droppedKey);
            if (askUserToEquip()) {
                player.inventory.key = droppedKey;
            }
        }
    }
```

Testing

[198]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Player player = createNewPlayer("Asad", 15, 3, 1);
droppedItemSequence(player, gameInfo);
displayPlayerInventory(player);
```

```
-----------------
A Key Dropped!
Enchanted key
Success Chance: 70%
-----------------
Would you like to equip the item

 a

Invalid input(yes/no)
Would you like to equip the item

 1
```

```
Invalid input(yes/no)
Would you like to equip the item

 yes

-----------------
    Inventory
sword: empty
shield: empty
key: Enchanted key (70%)
-----------------
```

[197]:
```
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Player player = createNewPlayer("Asad", 15, 3, 1);
droppedItemSequence(player, gameInfo);
displayPlayerInventory(player);
```

```
-----------------
A Key Dropped!
Gold key
Success Chance: 43%
-----------------
Would you like to equip the item

 no

-----------------
    Inventory
sword: empty
shield: empty
key: empty
-----------------
```

2.7.45 looseItem

What it does Determines what item a player loses/ how much health is lost when caught in a trap #### Implementation (how it works) Displays the prompt on how they were caught. It then checks whether they have an item to be lost in the first place, if not they loose health (a random number between 1-(1 + 20%)). If they do, it looses an item.

[201]:
```
public static void looseItem(Player player, String prompt) {
        // determines what item a player looses/ calculates the amount of␣
 ↪health lost
        String lostItem = getRandomLostItemType();

        printString(prompt);

        if (lostItem.equals("sword") & player.inventory.sword != null) {
            printString("You loose your sword");
            player.inventory.sword = null;
        } else if (lostItem.equals("shield") & player.inventory.shield != null)␣
↪{
            printString("You loose your shield");
            player.inventory.shield = null;
        } else {
            int lostHealth = getRandomNum((player.health / 5) + 2) + 1;
            printString("You loose " + lostHealth + " health");
            player.health -= lostHealth;
        }
        player.inventory.key = null;
    }
```

Testing

[202]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
looseItem(player, "The room is trapped and you dont have a key!");
```

```
The room is trapped and you dont have a key!
You loose 4 health
```

[206]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Sword sword = createNewSword(gameInfo);
player.inventory.sword = sword;
looseItem(player, "The room is trapped and you dont have a key!");
```

```
The room is trapped and you dont have a key!
You loose your sword
```

2.7.46 trappedRoomSequence

What it does Is called when a room has a trap #### Implementation (how it works) Checks whether they have a key using an if statement If so then it checks whether it is successful If so then the key breaks and the sequence ends If not then the player loses an item or loses health as well as their key breaking If they do not have a key it checks the player loses an item or loses health

[207]:
```
public static void trappedRoomSequence(Player player) {
        // checks whether the player passes a trapped room
        printString("----------------");
        if (player.inventory.key == null) {
            looseItem(player, "The room is trapped and you dont have a key!");
        } else if (!determineIfKeyWorks(player.inventory.key)) {
            looseItem(player, "The room is trapped and your key didnt work!");
        } else {
            printString("The room is trapped but you successfully disarm the␣
↪trap!");
            player.inventory.key = null;
        }
        printString("----------------");
    }
```

Testing

[218]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Sword sword = createNewSword(gameInfo);
player.inventory.sword = sword;
trappedRoomSequence(player);
```

```
----------------
The room is trapped and you dont have a key!
You loose your sword
----------------
```

[220]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
trappedRoomSequence(player);
```

```
----------------
The room is trapped and you dont have a key!
You loose 5 health
----------------
```

[230]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Key key = createNewKey();
player.inventory.key = key;
trappedRoomSequence(player);
```

```
----------------
The room is trapped and your key didnt work!
You loose 5 health
----------------
```

[221]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Key key = createNewKey();
player.inventory.key = key;
trappedRoomSequence(player);
```

```
----------------
The room is trapped but you successfully disarm the trap!
----------------
```

2.7.47 playerDamage

What it does Calculates the total damage of a player and returns the total #### Implementation (how it works) Checks whether the player has a sword If so then it adds it to the total Damage as well as decrementing the durability If the durability is 0 after the fact, it breaks and becomes null Then returns the total If no sword then it just returns the players base damage

[231]:
```
public static int playerDamage(Player player) {
        // calculates the total damage of a player in a round, also updates␣
 ↪durability
        if (player.inventory.sword == null) {
            return player.damage;
        } else {
            int totalDamage = player.damage + player.inventory.sword.damage;
            player.inventory.sword.durability -= 1;
            if (player.inventory.sword.durability == 0) {
                printString("Your " + player.inventory.sword.name + " broke!");
                player.inventory.sword = null;
            }
            return totalDamage;
        }
    }
```

Testing

[235]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Sword sword = createNewSword(gameInfo);
player.inventory.sword = sword;
displayHud(gameInfo, player);
displayPlayerStats(player);
playerDamage(player);
```

```
-----------------
Score: 0
-----------------
-----------------
    Inventory
sword: Dagger (+ 2) 4
shield: empty
key: empty
-----------------
-----------------
name: Asad
health: 15
damage: 3
defence: 1
-----------------
```

[235]: 5

2.7.48 playerDefence

What it does Calculates the total defence of a player and returns the total #### Implementation (how it works) Checks whether the player has a shield If so then it adds it to the total Defence as well as decrementing the durability If the durability is 0 after the fact, it breaks and becomes null Then returns the total If no shield then it just returns the players base defence

[236]:
```
public static int playerDefence(Player player) {
        // calculates the total defence of a player in a round, also updates␣
 ↪durability
        if (player.inventory.shield == null) {
            return player.defence;
        } else {
            int totalDefence = player.defence + player.inventory.shield.defence;
            player.inventory.shield.durability -= 1;
            if (player.inventory.shield.durability == 0) {
                printString("Your " + player.inventory.shield.name + " broke!");
                player.inventory.shield = null;
            }
            return totalDefence;
        }
    }
```

Testing

[237]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Shield shield = createNewShield(gameInfo);
player.inventory.shield = shield;
displayHud(gameInfo, player);
displayPlayerStats(player);
playerDefence(player);
```

```
-----------------
Score: 0
-----------------
-----------------
    Inventory
sword: empty
shield: Enchanted Shield (+ 1) 1
key: empty
-----------------
-----------------
name: Asad
health: 15
damage: 3
```

```
defence: 1
-----------------
Your Enchanted Shield broke!
```

[237]:
```
2
```

2.7.49 healthLost

What it does Calculates the amount of health lost #### Implementation (how it works) Takes a damage and defence stat and subtracts the defence from the damage. If the defence >= the damage then it just returns 0

[238]:
```java
public static int healthLost(int damage, int defence) {
        // calculates the amount of health lost, if def > dmg then it returns 0
        if (damage <= defence) {
            return 0;
        } else {
            return damage - defence;
        }
    }
```

Testing

[240]:
```java
healthLost(10, 5);
```

[240]:
```
5
```

[241]:
```java
healthLost(5, 5);
```

[241]:
```
0
```

[242]:
```java
healthLost(4, 5);
```

[242]:
```
0
```

2.7.50 fightSequence

What it does Where the fight sequence occurs #### Implementation (how it works) Stores the number of total enemies in the room in totalEnemies While there are > 0 enemies

It creates an enemy based on the gameInfo variable. Also creates a round variable to store the number of rounds that have passed during the fight.

While the player health and the enemy health are > 0

it increases the Difficulty of the enemy every 5 rounds. Display the enemy stats and looses health based on the players total damage. Display the player stats and looses health based on the enemy's total damage. An item dropped sequence can occur. Finally the round increments.

if the players health < 0 then it returns false. if the enemy's health < 0 then the number of enemies decrement.

If the while loops both break then it returns true as the player survived the fight(s).

[243]:
```
public static boolean fightSequence(Room currentRoom, Player player, GameInfo␣
↪gameInfo) {
        // carries out the fight phase of the room
        int totalEnemies = currentRoom.enemies;
        printString("-----------------");
        while (currentRoom.enemies > 0) {
            printString("-Enemy Encoutered-");
            Enemy enemy = createNewEnemy(gameInfo);
            int round = 1;
            while (player.health > 0 & enemy.health > 0) {
                if (round % 6 == 0) {
                    increaseEnemyDifficulty(enemy);
                }
                printString("(" + currentRoom.enemies + "/" + totalEnemies +␣
↪")");

                displayEnemy(enemy);
                printString("        vs.");
                displayPlayerStats(player);
                int enemyHealthLost = healthLost(playerDamage(player), enemy.
↪defence);
                enemy.health -= enemyHealthLost;
                printString("You dealt " + enemyHealthLost + " damage!");

                int playerHealthLost = healthLost(enemy.damage,␣
↪playerDefence(player));
                player.health -= playerHealthLost;
                printString("You lost " + playerHealthLost + " health!");

                if (itemDropped()) {
                    droppedItemSequence(player, gameInfo);
                }

                pressEnterToContinue();
                clearScreen();
                displayHud(gameInfo, player);
                round++;
            }

            if (player.health <= 0) {
                return false;
            } else if (enemy.health <= 0) {
                currentRoom.enemies -= 1;
                printString("You killed the " + enemy.name + "!");
                printString("------------------");
                pressEnterToContinue();
                clearScreen();
                displayHud(gameInfo, player);
            }

        }

        return true;
    }
```

Testing

[245]:
```
Player player = createNewPlayer("Asad", 15, 50, 50);
// for testing purposes the damage will be abnormally large
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Room currentRoom = createNewRoom();
fightSequence(currentRoom, player, gameInfo)
```

```
-----------------
-Enemy Encoutered-
(1/1)
-----------------
name: Skeleton
health: 2
damage: 1
defence: 0
-----------------
        vs.
-----------------
name: Asad
health: 15
damage: 50
defence: 50
-----------------
You dealt 50 damage!
You lost 0 health!
Press Enter To Continue




-----------------
Score: 0
-----------------
-----------------
    Inventory
sword: empty
```

```
shield: empty
key: empty
-----------------
You killed the Skeleton!
-----------------
Press Enter To Continue




-----------------
Score: 0
-----------------
-----------------
    Inventory
sword: empty
shield: empty
key: empty
-----------------
```

[245]: true

[246]:
```
Player player = createNewPlayer("Asad", 1, 0, 0);
// for testing purposes the player stats will be very low
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Room currentRoom = createNewRoom();
fightSequence(currentRoom, player, gameInfo)
```

```
-----------------
-Enemy Encoutered-
(2/2)
-----------------
name: Slime
health: 3
damage: 1
defence: 0
-----------------
        vs.
-----------------
name: Asad
health: 1
damage: 0
defence: 0
-----------------
You dealt 0 damage!
You lost 1 health!
Press Enter To Continue
```

```
----------------
Score: 0
----------------
----------------
    Inventory
sword: empty
shield: empty
key: empty
----------------
```

[246]:
```
false
```

2.7.51 saveGame

What it does Takes game information and fileName and saves the game into a .txt file #### Implementation (how it works) Uses the FileWriter class to open the file and write down the data stored in the variables passed into it. Its in a try and catch block to avoid crashing if the file is not found

[250]:
```
public static void saveGame(GameInfo gameInfo, Player player, String fileName) {
        // writes the game data to a save file
        try {
            FileWriter writer = new FileWriter(fileName + ".txt");
            writer.write("GameInfo" + "\n");
            writer.write(gameInfo.maximumHealth + "\n");
            writer.write(gameInfo.maximumDamage + "\n");
            writer.write(gameInfo.maximumDefence + "\n");
            writer.write(gameInfo.rooms + "\n");
            if (gameInfo.gameover) {
                writer.write("true" + "\n");
            } else {
                writer.write("false" + "\n");
            }

            writer.write("Player" + "\n");
            writer.write(player.name + "\n");
            writer.write(player.health + "\n");
            writer.write(player.damage + "\n");
            writer.write(player.defence + "\n");

            if (!(player.inventory.sword == null)) {
                writer.write("Sword" + "\n");
                writer.write(player.inventory.sword.name + "\n");
                writer.write(player.inventory.sword.durability + "\n");
                writer.write(player.inventory.sword.damage + "\n");
                writer.write("-----" + "\n");
            }

            if (!(player.inventory.shield == null)) {
                writer.write("Shield" + "\n");
                writer.write(player.inventory.shield.name + "\n");
                writer.write(player.inventory.shield.durability + "\n");
                writer.write(player.inventory.shield.defence + "\n");
                writer.write("------" + "\n");
            }

            if (!(player.inventory.key == null)) {
                writer.write("Key" + "\n");
                writer.write(player.inventory.key.name + "\n");
                writer.write(player.inventory.key.disarmChance + "\n");
                writer.write("---" + "\n");
            }

            writer.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

Testing

[248]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
saveGame(gameInfo, player, "game");
```

This is what will be saved on the .txt fileGameInfo 10 2 1 0 false Player Asad 15 3 1

[251]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
Shield shield = createNewShield(gameInfo);
player.inventory.shield = shield;
saveGame(gameInfo, player, "game1");
```

This is what will be saved on the .txt fileGameInfo 10 2 1 0 false Player Asad 15 3 1 Shield Steel Plated Shield 3 2 ——

2.7.52 readSaveFile

What it does Takes a .txt and loads a game #### Implementation (how it works) Reads each line of a .txt files and loads it into its corresponding records. In a try and catch block to avoid crashing if a file is not found. Passes the new records into a loadGame function and the game continues. If the game has a gameover value of false however, the game does not load and prompts the user “Save file is already game over”.

[269]:
```
public static boolean readSaveFile(String fileName) {
        try {
            File file = new File(fileName + ".txt");
            Scanner reader = new Scanner(file);
            while (reader.hasNextLine()) {
                String data = reader.nextLine();
                if (data.equals("true")) {
                    printString("Save file is already game over");
                    return false;
                }
            }
            reader.close();

            reader = new Scanner(file);

            String data = reader.nextLine();
            int[] dataArray = new int[4];
            for (int i = 0; i < 4; i++) {
                data = reader.nextLine();
                dataArray[i] = Integer.parseInt(data);
            }
            GameInfo gameInfo = createNewGameInfo(dataArray[0], dataArray[1],␣
↪dataArray[2], dataArray[3]);
            data = reader.nextLine();
            data = reader.nextLine();

            dataArray = new int[3];
            String name = reader.nextLine();
            for (int i = 0; i < 3; i++) {
                data = reader.nextLine();
                dataArray[i] = Integer.parseInt(data);
            }
            Player player = createNewPlayer(name, dataArray[0], dataArray[1],␣
↪dataArray[2]);
            data = reader.nextLine();

            if (data.equals("Sword")) {
                dataArray = new int[2];
                String swordName = reader.nextLine();
                for (int i = 0; i < 2; i++) {
                    data = reader.nextLine();
                    dataArray[i] = Integer.parseInt(data);
                }
                Sword sword = loadSword(swordName, dataArray);
```

```java
                player.inventory.sword = sword;
                data = reader.nextLine();
            }

            if (data.equals("Shield")) {
                dataArray = new int[2];
                String shieldName = reader.nextLine();
                for (int i = 0; i < 2; i++) {
                    data = reader.nextLine();
                    dataArray[i] = Integer.parseInt(data);
                }
                Shield shield = loadShield(shieldName, dataArray);
                player.inventory.shield = shield;
                data = reader.nextLine();
            }

            if (data.equals("Key")) {
                String keyName = reader.nextLine();
                int disarmChance = Integer.parseInt(reader.nextLine());

                Key key = loadKey(keyName, disarmChance);
                player.inventory.key = key;
                data = reader.nextLine();
            }
            loadGame(gameInfo, player, fileName);

        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
            return false;
        }

        return true;
    }
```

Testing

2.8 game1.txt

GameInfo 10 2 1 0 false Player Asad 15 3 1 Shield Steel Plated Shield 3 2 ——

[297]:
```java
readSaveFile("game1"); // interuppt for testing purposes only
```

```
-----------------
Score: 0
-----------------
-----------------
```

```
    Inventory
sword: empty
shield: Steel Plated Shield (+ 2) 3
key: empty
-----------------
-----------------
The room is trapped and you dont have a key!
You loose 4 health
-----------------
-----------------
-Enemy Encoutered-
(1/1)
-----------------
name: Witch
health: 6
damage: 1
defence: 0
-----------------
        vs.
-----------------
name: Asad
health: 11
damage: 3
defence: 1
-----------------
You dealt 3 damage!
You lost 0 health!
-----------------
Some food Dropped!
You ate the Apple pie and healed 1 points.
-----------------
Press Enter To Continue
```

```
org.zeromq.ZMQException: Errno 4 : errno 4
        at org.zeromq.ZMQ$Socket.mayRaise(ZMQ.java:3533)
        at org.zeromq.ZMQ$Socket.recv(ZMQ.java:3364)
        at org.zeromq.ZMQ$Socket.recv(ZMQ.java:3339)
```

```
|   readSaveFile("game1");
Evaluation interrupted.
```

```
        at io.github.spencerpark.jupyter.channels.JupyterSocket.readMessage(Jupy
terSocket.java:74)
```

2.9 game2.txt

GameInfo 14 4 1 2 true Player Asad 12 5 1

[304]:
```
readSaveFile("game2");
```

```
Save file is already game over
```

[304]:
```
false
```

2.9.1 loadGame

What it does Where the main game loop exists #### Implementation (how it works) Requires a GameInfo Player and a filename variable to work

Whilst the game isn't over the loop will keep passing

Saves the game each room and displays the HUD Carries out a dropped item sequence if item dropped

Creates a new room using createNewRoom

If the room is trapped it carries out a trapped room sequence If the player survives the trap it then carries out the Fight sequence If the player doesn't survive the trap gameInfo.gameover is set to true

If the player wins the fight and the game isn't already over it increases the game difficulty aswell as boost the players stats and flushes the screen If the player loses the fight and the game wasn't already over it displays an appropriate game over message and gameInfo.gameover is set to true

when the loop breaks it displays their final score and saves the games final information

[191]:
```
public static void loadGame(GameInfo gameInfo, Player player, String fileName) {
        while (gameInfo.gameover == false) {
            saveGame(gameInfo, player, fileName);
            displayHud(gameInfo, player);

            if (itemDropped()) {
                droppedItemSequence(player, gameInfo);
            }

            Room currentRoom = createNewRoom();

            if (currentRoom.isTrap) {
                trappedRoomSequence(player);
                if (player.health <= 0) {
                    gameOverMessage("You died to the trap!");
                    gameInfo.gameover = true;
                }
            }
            boolean fightWon = fightSequence(currentRoom, player, gameInfo);

            if (fightWon & !(gameInfo.gameover)) {
                increaseGameDifficulty(gameInfo);
                increaseBaseStats(player);
                pressEnterToContinue();
                clearScreen();
            } else if (!fightWon & !(gameInfo.gameover)) {
                gameOverMessage("You died in the fight!");
                gameInfo.gameover = true;
            }

        }
        printString("Your final score: " + gameInfo.rooms);
        saveGame(gameInfo, player, fileName);
    }
```

Testing

[303]:
```
Player player = createNewPlayer("Asad", 15, 3, 1);
GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
loadGame(gameInfo, player, "testGame"); // interuppt for testing purposes only
```

```
-----------------
Score: 0
-----------------
-----------------
    Inventory
sword: empty
shield: empty
key: empty
-----------------
-----------------
-Enemy Encoutered-
(1/1)
-----------------
name: Slime
health: 9
damage: 1
defence: 0
-----------------
        vs.
-----------------
name: Asad
health: 15
damage: 3
defence: 1
-----------------
```

```
You dealt 3 damage!
You lost 0 health!
Press Enter To Continue


org.zeromq.ZMQException: Errno 4 : errno 4
        at org.zeromq.ZMQ$Socket.mayRaise(ZMQ.java:3533)
        at org.zeromq.ZMQ$Socket.recv(ZMQ.java:3364)
```

```
|   loadGame(gameInfo, player, "testGame");
Evaluation interrupted.
```

```
        at org.zeromq.ZMQ$Socket.recv(ZMQ.java:3339)
```

2.9.2 startNewGame

What it does Welcomes the player to a new game and loads them into a game #### Implementation (how it works) Asks the player for their name using askPlayerName and the name of the save file. Creates a new GameInfo variable and a new Player variable. The data is then passed into the loadGame function

[299]:
```
public static void startNewGame() {
        // where the main game loop takes place
        printString("Welcome to booby trap castle");
        String name = askPlayerName();
        GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
        Player player = createNewPlayer(name, 15, 3, 1);
        String fileName = askFileName();
        printString("Hello  " + player.name + "!");
        printString("Get through as many rooms as possible!");
        printString("");

        loadGame(gameInfo, player, fileName);
    }
```

Testing

[301]:
```
startNewGame(); // interuppt for testing purposes only
```

```
Welcome to booby trap castle
What is your name:

 Asad

Save file name:

 newGame
```

```
Hello  Asad!
Get through as many rooms as possible!

-----------------
Score: 0
-----------------
-----------------
    Inventory
sword: empty
shield: empty
key: empty
-----------------
-----------------
A Shield Dropped!
Enchanted Shield (+ 2) 2
-----------------
Would you like to equip the item
```

```
|   startNewGame();
Evaluation interrupted.
```

2.9.3 mainMenu

What it does Carries out the main menu sequence #### Implementation (how it works) Displays the options and takes an input using the displayMainMenu and askMenuOption functions. Uses a if statement to determine which function to call

[189]:
```
public static void mainMenu() {
        boolean option;
        displayMainMenu();
        option = askMenuOption();
        if (option) {
            startNewGame();
        } else {
            String fileName = askFileName();
            if(!readSaveFile(fileName)){
                printString("");
            }
        }
    }
```

Testing

[306]:
```
mainMenu(); // interuppt for testing purposes only
```

```
-----------------
Menu
(1) Start new game
(2) Load game
-----------------
Select your option

 1

Welcome to booby trap castle
What is your name:

 Asad

Save file name:

 game3

Hello  Asad!
Get through as many rooms as possible!

-----------------
Score: 0
-----------------
-----------------
    Inventory
sword: empty
shield: empty
key: empty
-----------------
-----------------
-Enemy Encoutered-
(1/1)
-----------------
name: Zombie
health: 1
damage: 2
defence: 0
-----------------
        vs.
-----------------
name: Asad
health: 15
damage: 3
defence: 1
-----------------
You dealt 3 damage!
You lost 1 health!
Press Enter To Continue

org.zeromq.ZMQException: Errno 4 : errno 4
```

```
|   mainMenu();
Evaluation interrupted.
```

```
        at org.zeromq.ZMQ$Socket.mayRaise(ZMQ.java:3533)        at
org.zeromq.ZMQ$Socket.recv(ZMQ.java:3364)
```

[308]:
```
mainMenu(); // interuppt for testing purposes only
```

```
-----------------
Menu
(1) Start new game
(2) Load game
-----------------
Select your option

 game1

Invalid input(1/2)
Select your option

 2

Save file name:

 game1

-----------------
Score: 1
-----------------
-----------------
    Inventory
sword: empty
shield: Steel Plated Shield (+ 2) 1
key: empty
-----------------
-----------------
Some food Dropped!
You ate the Beef Jerky and healed 1 points.
-----------------
-----------------
-Enemy Encoutered-
(1/1)
-----------------
name: Goblin
health: 2
damage: 1
defence: 0
-----------------
```

```
        vs.
-----------------
name: Asad
health: 13
damage: 4
defence: 1
-----------------
You dealt 4 damage!
Your Steel Plated Shield broke!
You lost 0 health!
Press Enter To Continue
```

```
org.zeromq.ZMQException: Errno 4 : errno 4
```

```
|   mainMenu(); // interuppt for testing purposes only
Evaluation interrupted.
```

```
        at org.zeromq.ZMQ$Socket.mayRaise(ZMQ.java:3533)
```

2.9.4 Running the program

Run the following call to simulate running the complete program.

[309]:
```
mainMenu(); // interuppt for testing purposes only
```

```
-----------------
Menu
(1) Start new game
(2) Load game
-----------------
Select your option

 1

Welcome to booby trap castle
What is your name:

 Asad

Save file name:

 finalGameTest

Hello  Asad!
Get through as many rooms as possible!

-----------------
Score: 0
-----------------
-----------------
```

```
    Inventory
sword: empty
shield: empty
key: empty
----------------
----------------
-Enemy Encoutered-
(1/1)
----------------
name: Ogre
health: 4
damage: 1
defence: 0
----------------
        vs.
----------------
name: Asad
health: 15
damage: 3
defence: 1
----------------
You dealt 3 damage!
You lost 0 health!
Press Enter To Continue




----------------
Score: 0
----------------
----------------
    Inventory
sword: empty
shield: empty
key: empty
----------------
(1/1)
----------------
name: Ogre
health: 1
damage: 1
defence: 0
----------------
        vs.
----------------
name: Asad
health: 15
```

```
damage: 3
defence: 1
-----------------
You dealt 3 damage!
You lost 0 health!
Press Enter To Continue




-----------------
Score: 0
-----------------
-----------------
    Inventory
sword: empty
shield: empty
key: empty
-----------------
You killed the Ogre!
-----------------
Press Enter To Continue




-----------------
Score: 0
-----------------
-----------------
    Inventory
sword: empty
shield: empty
key: empty
-----------------
-----------------
Your defence increased
-----------------
Press Enter To Continue
```

```
org.zeromq.ZMQException: Errno 4 : errno 4
        at org.zeromq.ZMQ$Socket.mayRaise(ZMQ.java:3533)
        at org.zeromq.ZMQ$Socket.recv(ZMQ.java:3364)
        at org.zeromq.ZMQ$Socket.recv(ZMQ.java:3339)
        at io.github.spencerpark.jupyter.channels.JupyterSocket.readMessage(Jupy
terSocket.java:74)
        at io.github.spencerpark.jupyter.channels.JupyterSocket.readMessage(Jupy
terSocket.java:118)
```

```
        at io.github.spencerpark.jupyter.channels.StdinChannel.getInput(StdinCha
nnel.java:47)
        at io.github.spencerpark.jupyter.channels.ShellReplyEnvironment.readFrom
StdIn(ShellReplyEnvironment.java:39)
```

```
|   mainMenu();
Evaluation interrupted.
```

```
        at io.github.spencerpark.jupyter.channels.ShellReplyEnvironment.readFrom
StdIn(ShellReplyEnvironment.java:47)
        at io.github.spencerpark.jupyter.channels.JupyterInputStream.readFromFro
ntend(JupyterInputStream.java:57)
```

2.10 The complete program

This version will only compile here. To run it copy it into a file called initials.java on your local computer and compile and run it there.

[310]:
```java
/* ***************************************
  @author    Asad Ali Khan
  @SID       220257466
  @version   1

    Explore the never ending castle - game
   ****************************************/

import java.util.Random;
import java.util.Scanner;
import java.io.FileWriter;
import java.io.IOException;
import java.io.File;
import java.io.FileNotFoundException;

class Game {

    public static void main(String[] args) {
        mainMenu();
    }

    public static void printString(String string) {
        // prints the string that is passed into it
        System.out.println(string);
    }

    public static void pressEnterToContinue() {
        // waits for the user to enter a new line to continue to the next action
        printString("Press Enter To Continue");
        Scanner scanner = new Scanner(System.in);
        try {
            System.in.read();
            scanner.nextLine();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void clearScreen() {
        // flushes the screen of its text to make the game easier to read
        printString("\033[H\033[2J");
    }

    public static void startNewGame() {
        // where the main game loop takes place
        printString("Welcome to booby trap castle");
        String name = askPlayerName();
        GameInfo gameInfo = createNewGameInfo(10, 2, 1, 0);
        Player player = createNewPlayer(name, 15, 3, 1);
        String fileName = askFileName();
        printString("Hello  " + player.name + "!");
        printString("Get through as many rooms as possible!");
        printString("");

        loadGame(gameInfo, player, fileName);
    }

    public static void loadGame(GameInfo gameInfo, Player player, String␣
↪fileName) {
        while (gameInfo.gameover == false) {
            saveGame(gameInfo, player, fileName);
            displayHud(gameInfo, player);

            if (itemDropped()) {
                droppedItemSequence(player, gameInfo);
            }

            Room currentRoom = createNewRoom();

            if (currentRoom.isTrap) {
                trappedRoomSequence(player);
                if (player.health <= 0) {
                    gameOverMessage("You died to the trap!");
                    gameInfo.gameover = true;
                }
            }

            boolean fightWon = fightSequence(currentRoom, player, gameInfo);

            if (fightWon & !(gameInfo.gameover)) {
                increaseGameDifficulty(gameInfo);
                increaseBaseStats(player);
                pressEnterToContinue();
                clearScreen();
            } else if (!fightWon & !(gameInfo.gameover)) {
                gameOverMessage("You died in the fight!");
                gameInfo.gameover = true;
            }

        }
        printString("Your final score: " + gameInfo.rooms);
        saveGame(gameInfo, player, fileName);
    }

    public static String askFileName() {
        // asks the suer to enter a file and returns the filename
        String fileName;
        Scanner userinput = new Scanner(System.in);

        printString("Save file name: ");
        fileName = userinput.nextLine();

        return fileName;

    }

    public static String askPlayerName() {
        // takes a user input for the players name and returns it
        String name;
        Scanner userinput = new Scanner(System.in);

        printString("What is your name: ");
        name = userinput.nextLine();

        return name;

    }

    public static GameInfo createNewGameInfo(int maximumHealth, int␣
↪maximumDamage, int maximumDefence, int rooms) {
        // creates a new gameInfo variable
        GameInfo gameInfo = new GameInfo();
        gameInfo.gameover = false;

        gameInfo.rooms = rooms;
        gameInfo.maximumHealth = maximumHealth;
        gameInfo.maximumDamage = maximumDamage;
        gameInfo.maximumDefence = maximumDefence;

        return gameInfo;
    }

    public static void increaseGameDifficulty(GameInfo gameInfo) {
        // takes the gameInfo and increases its difficulty and increments the␣
↪room
        gameInfo.rooms += 1;
        gameInfo.maximumHealth += 2;
        if (getRandomNum(2) == 0) {
            gameInfo.maximumDamage += 1;
        } else {
            gameInfo.maximumDefence += 1;
        }

    }

    public static void increaseEnemyDifficulty(Enemy enemy) {
        // takes the enemy and increases its difficulty
        printString(enemy.name + " gets stronger!");
        enemy.health += getRandomNum((enemy.health / 5) + 2);
        enemy.damage += getRandomNum((enemy.damage / 5) + 2);
        enemy.defence += getRandomNum((enemy.defence / 5) + 2);
    }

    public static void increaseBaseStats(Player player) {
        // rolls a 2 sided dice to determine what type of item was dropped
        int numberRolled = getRandomNum(2);
        printString("--------~~~~-----");
        if (numberRolled == 0) {
            printString("Your damage increased");
            player.damage += 1;
        } else {
            printString("Your defence increased");
            player.defence += 1;
        }
        printString("--------~~~~-----");
    }

    public static Room createNewRoom() {
```

```
        // this creates a new room with the number of enemies specified
        Room room = new Room();
        room.enemies = getRandomNum(2) + 1;
        room.isTrap = determineIfTrappedRoom();
        return room;
    }

    public static Boolean determineIfTrappedRoom() {
        // this rolls a 5 sided dice, if the rolled number is 0 then the next␣
↪room is a
        // trap
        int numberRolled = getRandomNum(5);

        if (numberRolled == 0) {
            return true;
        } else {
            return false;
        }
    }

    public static Boolean determineIfKeyWorks(Key key) {
        // compares the success rate with a random number between 0-100, if its␣
↪more
        // then the key works
        int numberRolled = getRandomNum(99) + 1;

        if (numberRolled <= key.disarmChance) {
            return true;
        } else {
            return false;
        }
    }

    public static Player createNewPlayer(String name, int health, int damage,␣
↪int defence) {
        // Creates a new player
        Player player = new Player();
        player.name = name;
        player.health = health;
        player.damage = damage;
        player.defence = defence;

        return player;
    }

    public static Enemy createNewEnemy(GameInfo gameInfo) {
        // this creates a new enemy
        Enemy enemy = new Enemy();

        String[] names = {
                "Slime",
                "Zombie",
                "Skeleton",
                "Ogre",
                "Troll",
                "Goblin",
                "Witch",
        };

        enemy.name = names[getRandomNum(names.length)];
        enemy.health = getRandomNum(gameInfo.maximumHealth) + 1;
        enemy.damage = getRandomNum(gameInfo.maximumDamage) + 1;
        enemy.defence = getRandomNum(gameInfo.maximumDefence);

        return enemy;
    }

    public static Sword createNewSword(GameInfo gameInfo) {
        // this creates a new sword
        Sword sword = new Sword();

        String[] names = {
                "Long Sword",
                "Dagger",
                "Short Sword",
                "Mace",
                "Katana",
                "Battle Axe",
        };

        sword.name = names[getRandomNum(names.length)];
        sword.durability = getRandomNum(7) + 1;
        sword.damage = getRandomNum((int) (gameInfo.maximumHealth / 5.0)) + 1;

        return sword;
    }

    public static Sword loadSword(String name, int[] dataArray) {
        Sword sword = new Sword();
        sword.name = name;
        sword.durability = dataArray[0];
        sword.damage = dataArray[1];

        return sword;
    }

    public static Shield createNewShield(GameInfo gameInfo) {
        // this creates a new shield
        Shield shield = new Shield();

        String[] names = {
                "Broad Shield",
                "Enchanted Shield",
                "Steel Plated Shield",
        };

        shield.name = names[getRandomNum(names.length)];
        shield.durability = getRandomNum((int) (3)) + 1;
        shield.defence = getRandomNum((int) (gameInfo.maximumHealth / 5.0)) + 1;

        return shield;
    }

    public static Shield loadShield(String name, int[] dataArray) {
        Shield shield = new Shield();
        shield.name = name;
        shield.durability = dataArray[0];
        shield.defence = dataArray[1];

        return shield;

    }

    public static Food createNewFood(GameInfo gameInfo) {
        // this creates a new food item
        Food food = new Food();

        String[] names = { "Beef Jerky", "Apple pie", "Hearty bread" };

        food.name = names[getRandomNum(names.length)];
        food.healthRegen = getRandomNum((int) (gameInfo.maximumHealth / 3.5)) +␣
↪1;

        return food;
    }

    public static Key createNewKey() {
        // this creates a new key
        Key key = new Key();
        String[] names = { "Gold key", "Old key", "Enchanted key" };

        key.name = names[getRandomNum(names.length)];
        key.disarmChance = getRandomNum(100) + 1;

        return key;
    }

    public static Key loadKey(String name, int disarmChance) {
        Key key = new Key();
        key.name = name;
        key.disarmChance = disarmChance;

        return key;

    }

    public static int getRandomNum(int num) {
        // rolls a dice with the number of sides passed to it
        Random random = new Random();
        int rngNum = random.nextInt(num);

        return rngNum;
    }

    public static boolean itemDropped() {
        // rolls a 5 sided dice to determine if an item was dropped
        int numberRolled = getRandomNum(5);

        if (numberRolled == 0) {
            return true;
        } else {
            return false;
        }
    }

    public static String getRandomDroppedItemType() {
        // rolls a 4 sided dice to determine what type of item was dropped
        int numberRolled = getRandomNum(4);

        if (numberRolled == 0) {
            return "sword";
        } else if (numberRolled == 1) {
            return "shield";
        } else if (numberRolled == 2) {
            return "food";
        } else {
            return "key";
        }
    }

    public static String getRandomLostItemType() {
        // rolls a 2 sided dice to determine what type of item was dropped
        int numberRolled = getRandomNum(2);

        if (numberRolled == 0) {
            return "sword";
        } else {
            return "shield";
        }
    }

    public static void displayMainMenu() {
        // displays the Main menu
        printString("----------------");
        printString("Menu");
        printString("(1) Start new game");
        printString("(2) Load game");
        printString("----------------");
    }

    public static void displayDroppedSword(Sword sword) {
        // displays the sword that is dropped
        printString("----------------");
        printString("A Sword Dropped!");
        printString(sword.name + " (+ " + sword.damage + ") " + sword.
↪durability);
        printString("----------------");
    }

    public static void displayDroppedShield(Shield shield) {
        // displays the shield that is dropped
        printString("----------------");
        printString("A Shield Dropped!");
        printString(shield.name + " (+ " + shield.defence + ") " + shield.
↪durability);
        printString("----------------");
    }

    public static void displayDroppedFood(Food food) {
        // displays the food that is dropped
        printString("----------------");
        printString("Some food Dropped!");
        printString(
                "You ate the " +
                        food.name +
                        " and healed " +
                        food.healthRegen +
                        " points.");
        printString("-------~~~-----");
    }

    public static void displayDroppedKey(Key key) {
        // displays the key that is dropped
        printString("-------~~~-----");
        printString("A Key Dropped!");
        printString(key.name);
        printString("Success Chance: " + key.disarmChance + "%");
        printString("-------~~~-----");
    }

    public static void displayScore(GameInfo gameInfo) {
        // displays the current room/score
        printString("-------~~~-----");
        printString("Score: " + gameInfo.rooms);
        printString("-------~~~-----");
    }

    public static void displayPlayerStats(Player player) {
        // displays the players stats
        printString("-------~~~-----");
        printString("name: " + player.name);
        printString("health: " + player.health);
        printString("damage: " + player.damage);
        printString("defence: " + player.defence);
        printString("-------~~~-----");
    }

    public static void displayPlayerInventory(Player player) {
        // displays the players inventory
        printString("-------~~~-----");
        printString("    Inventory    ");
        displaySword(player);
        displayShield(player);
        displayKey(player);
        printString("-------~~~-----");
    }

    public static void displaySword(Player player) {
        // displays an equipped sword
        if (player.inventory.sword == null) {
```

```
            printString("sword: empty");
        } else {
            printString("sword: " + player.inventory.sword.name + " (+ " +␣
↪player.inventory.sword.damage + ") "
                    + player.inventory.sword.durability);
        }
    }

    public static void displayShield(Player player) {
        // displays an equipped shield
        if (player.inventory.shield == null) {
            printString("shield: empty");
        } else {
            printString("shield: " + player.inventory.shield.name + " (+ " +␣
↪player.inventory.shield.defence + ") "
                    + player.inventory.shield.durability);
        }
    }

    public static void displayKey(Player player) {
        // displays an equipped shield
        if (player.inventory.key == null) {
            printString("key: empty");
        } else {
            printString("key: " + player.inventory.key.name + " (" + player.
↪inventory.key.disarmChance + "%)");
        }
    }

    public static void displayEnemy(Enemy enemy) {
        // displays an enemys stats
        printString("--------~~~-------");
        printString("name: " + enemy.name);
        printString("health: " + enemy.health);
        printString("damage: " + enemy.damage);
        printString("defence: " + enemy.defence);
        printString("--------~~~-------");
    }

    public static void displayHud(GameInfo gameInfo, Player player) {
        // displays the score and inventory together
        displayScore(gameInfo);
        displayPlayerInventory(player);
    }

    public static boolean askUserToEquip() {
        // asks the user whether they would like to equip the dropped item
```

```
        Scanner userinput = new Scanner(System.in);
        printString("Would you like to equip the item");
        String answer = userinput.nextLine();

        while (!answer.equals("yes") & !answer.equals("no")) {
            printString("Invalid input(yes/no)");
            printString("Would you like to equip the item");
            answer = userinput.nextLine();
        }

        if (answer.equals("yes")) {
            return true;
        } else {
            return false;
        }
    }

    public static boolean askMenuOption() {
        // asks the user what option they would like to proceed with
        Scanner userinput = new Scanner(System.in);
        printString("Select your option");
        String answer = userinput.nextLine();

        while (!answer.equals("1") & !answer.equals("2")) {
            printString("Invalid input(1/2)");
            printString("Select your option");
            answer = userinput.nextLine();
        }

        if (answer.equals("1")) {
            return true;
        } else {
            return false;
        }
    }

    public static void eatFood(Player player, Food food) {
        // players health increases with the food
        player.health += food.healthRegen;
    }

    public static void mainMenu() {
        boolean option;
        displayMainMenu();
        option = askMenuOption();
        if (option) {
            startNewGame();
```

```java
        } else {
            String fileName = askFileName();
            if(!readSaveFile(fileName)){
                printString("");
            }
        }
    }

    public static void droppedItemSequence(Player player, GameInfo gameInfo) {
        // asks the user whether they would like to equip the dropped item
        String droppedItemType = getRandomDroppedItemType();

        if (droppedItemType.equals("sword")) {
            Sword droppedSword = createNewSword(gameInfo);
            displayDroppedSword(droppedSword);
            if (askUserToEquip()) {
                player.inventory.sword = droppedSword;
            }
        } else if (droppedItemType.equals("shield")) {
            Shield droppedShield = createNewShield(gameInfo);
            displayDroppedShield(droppedShield);
            if (askUserToEquip()) {
                player.inventory.shield = droppedShield;
            }
        } else if (droppedItemType.equals("food")) {
            Food droppedFood = createNewFood(gameInfo);
            displayDroppedFood(droppedFood);
            eatFood(player, droppedFood);
        } else if (droppedItemType.equals("key")) {
            Key droppedKey = createNewKey();
            displayDroppedKey(droppedKey);
            if (askUserToEquip()) {
                player.inventory.key = droppedKey;
            }
        }
    }

    public static void looseItem(Player player, String prompt) {
        // determines what item a player looses/ calculates the amount of␣
↪health lost
        String lostItem = getRandomLostItemType();

        printString(prompt);

        if (lostItem.equals("sword") & player.inventory.sword != null) {
            printString("You loose your sword");
            player.inventory.sword = null;
        } else if (lostItem.equals("shield") & player.inventory.shield != null)␣
↪{
            printString("You loose your shield");
            player.inventory.shield = null;
        } else {
            int lostHealth = getRandomNum((player.health / 5) + 2) + 1;
            printString("You loose " + lostHealth + " health");
            player.health -= lostHealth;
        }
        player.inventory.key = null;
    }

    public static void trappedRoomSequence(Player player) {
        // checks whether the player passes a trapped room
        printString("------------------");
        if (player.inventory.key == null) {
            looseItem(player, "The room is trapped and you dont have a key!");
        } else if (!determineIfKeyWorks(player.inventory.key)) {
            looseItem(player, "The room is trapped and your key didnt work!");
        } else {
            printString("The room is trapped but you successfully disarm the␣
↪trap!");
            player.inventory.key = null;
        }
        printString("------------------");
    }

    public static int playerDamage(Player player) {
        // calculates the total damage of a player in a round, also updates␣
↪durability
        if (player.inventory.sword == null) {
            return player.damage;
        } else {
            int totalDamage = player.damage + player.inventory.sword.damage;
            player.inventory.sword.durability -= 1;
            if (player.inventory.sword.durability == 0) {
                printString("Your " + player.inventory.sword.name + " broke!");
                player.inventory.sword = null;
            }
            return totalDamage;
        }
    }

    public static int playerDefence(Player player) {
        // calculates the total defence of a player in a round, also updates␣
↪durability
        if (player.inventory.shield == null) {
            return player.defence;
        } else {
            int totalDefence = player.defence + player.inventory.shield.defence;
            player.inventory.shield.durability -= 1;
            if (player.inventory.shield.durability == 0) {
                printString("Your " + player.inventory.shield.name + " broke!");
                player.inventory.shield = null;
            }
            return totalDefence;
        }
    }

    public static int healthLost(int damage, int defence) {
        // calculates the amount of health lost, if def > dmg then it returns 0
        if (damage <= defence) {
            return 0;
        } else {
            return damage - defence;
        }
    }

    public static boolean fightSequence(Room currentRoom, Player player,␣
↪GameInfo gameInfo) {
        // carries out the fight phase of the room
        int totalEnemies = currentRoom.enemies;
        printString("--------~~~------");
        while (currentRoom.enemies > 0) {
            printString("-Enemy Encoutered-");
            Enemy enemy = createNewEnemy(gameInfo);
            int round = 1;
            while (player.health > 0 & enemy.health > 0) {
                if (round % 6 == 0) {
                    increaseEnemyDifficulty(enemy);
                }
                printString("(" + currentRoom.enemies + "/" + totalEnemies +␣
↪")");
                displayEnemy(enemy);
                printString("        vs.");
                displayPlayerStats(player);
                int enemyHealthLost = healthLost(playerDamage(player), enemy.
↪defence);
                enemy.health -= enemyHealthLost;
                printString("You dealt " + enemyHealthLost + " damage!");

                int playerHealthLost = healthLost(enemy.damage,␣
↪playerDefence(player));
                player.health -= playerHealthLost;
```

```
                printString("You lost " + playerHealthLost + " health!");

                if (itemDropped()) {
                    droppedItemSequence(player, gameInfo);
                }

                pressEnterToContinue();
                clearScreen();
                displayHud(gameInfo, player);
                round++;
            }

            if (player.health <= 0) {
                return false;
            } else if (enemy.health <= 0) {
                currentRoom.enemies -= 1;
                printString("You killed the " + enemy.name + "!");
                printString("--------~~~------");
                pressEnterToContinue();
                clearScreen();
                displayHud(gameInfo, player);
            }

        }

        return true;
    }

    public static void gameOverMessage(String prompt) {
        // prints out a gameover message
        printString("-=-=-=-=-=-=-=-=-=-");
        printString(prompt);
        printString("-=-=-=-=-=-=-=-=-=-");
    }

    public static void saveGame(GameInfo gameInfo, Player player, String␣
↪fileName) {
        // writes the game data to a save file
        try {
            FileWriter writer = new FileWriter(fileName + ".txt");
            writer.write("GameInfo" + "\n");
            writer.write(gameInfo.maximumHealth + "\n");
            writer.write(gameInfo.maximumDamage + "\n");
            writer.write(gameInfo.maximumDefence + "\n");
            writer.write(gameInfo.rooms + "\n");
            if (gameInfo.gameover) {
                writer.write("true" + "\n");
```

```
            } else {
                writer.write("false" + "\n");
            }

            writer.write("Player" + "\n");
            writer.write(player.name + "\n");
            writer.write(player.health + "\n");
            writer.write(player.damage + "\n");
            writer.write(player.defence + "\n");

            if (!(player.inventory.sword == null)) {
                writer.write("Sword" + "\n");
                writer.write(player.inventory.sword.name + "\n");
                writer.write(player.inventory.sword.durability + "\n");
                writer.write(player.inventory.sword.damage + "\n");
                writer.write("-----" + "\n");
            }

            if (!(player.inventory.shield == null)) {
                writer.write("Shield" + "\n");
                writer.write(player.inventory.shield.name + "\n");
                writer.write(player.inventory.shield.durability + "\n");
                writer.write(player.inventory.shield.defence + "\n");
                writer.write("------" + "\n");
            }

            if (!(player.inventory.key == null)) {
                writer.write("Key" + "\n");
                writer.write(player.inventory.key.name + "\n");
                writer.write(player.inventory.key.disarmChance + "\n");
                writer.write("---" + "\n");
            }

            writer.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static boolean readSaveFile(String fileName) {
        try {
            File file = new File(fileName + ".txt");
            Scanner reader = new Scanner(file);
            while (reader.hasNextLine()) {
                String data = reader.nextLine();
                if (data.equals("true")) {
```

```
                printString("Save file is already game over");
                return false;
            }
        }
        reader.close();

        reader = new Scanner(file);

        String data = reader.nextLine();
        int[] dataArray = new int[4];
        for (int i = 0; i < 4; i++) {
            data = reader.nextLine();
            dataArray[i] = Integer.parseInt(data);
        }
        GameInfo gameInfo = createNewGameInfo(dataArray[0], dataArray[1], 
↪dataArray[2], dataArray[3]);
        data = reader.nextLine();
        data = reader.nextLine();

        dataArray = new int[3];
        String name = reader.nextLine();
        for (int i = 0; i < 3; i++) {
            data = reader.nextLine();
            dataArray[i] = Integer.parseInt(data);
        }
        Player player = createNewPlayer(name, dataArray[0], dataArray[1], 
↪dataArray[2]);
        data = reader.nextLine();

        if (data.equals("Sword")) {
            dataArray = new int[2];
            String swordName = reader.nextLine();
            for (int i = 0; i < 2; i++) {
                data = reader.nextLine();
                dataArray[i] = Integer.parseInt(data);
            }
            Sword sword = loadSword(swordName, dataArray);
            player.inventory.sword = sword;
            data = reader.nextLine();
        }

        if (data.equals("Shield")) {
            dataArray = new int[2];
            String shieldName = reader.nextLine();
            for (int i = 0; i < 2; i++) {
                data = reader.nextLine();
                dataArray[i] = Integer.parseInt(data);
```

```java
                }
                Shield shield = loadShield(shieldName, dataArray);
                player.inventory.shield = shield;
                data = reader.nextLine();
            }

            if (data.equals("Key")) {
                String keyName = reader.nextLine();
                int disarmChance = Integer.parseInt(reader.nextLine());

                Key key = loadKey(keyName, disarmChance);
                player.inventory.key = key;
                data = reader.nextLine();
            }
            loadGame(gameInfo, player, fileName);

        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
            return false;
        }

        return true;
    }
}

class Player {

    String name;
    int health;
    int damage;
    int defence;
    Inventory inventory = new Inventory();
}

class Inventory {

    Sword sword;
    Shield shield;
    Key key;
}

class Enemy {

    String name;
    int health;
    int damage;
    int defence;
}

class Sword {

    String name;
    int durability;
    int damage;
}

class Shield {

    String name;
    int durability;
    int defence;
}

class Food {

    String name;
    int healthRegen;
}

class Key {

    String name;
    int disarmChance;
}

class Room {

    int enemies;
    boolean isTrap;
}

class GameInfo {

    int maximumHealth; // minium health of spawned enemies
    int maximumDamage; // minimum damage of spawned enemies
    int maximumDefence; // maximum defence of spawned enemies
    int rooms; // stores the number of rooms that the player successfully passed
    boolean gameover; // stores wether the game is over or not
}
```

END OF LITERATE DOCUMENT