# 

# C:\Program Files (x86)\Microsoft Office\MEDIA\CAGCAT10\j0205466.wmf

[Introdution 4](#_Toc504319378)

[Syfte (Behövs syfte??) 4](#_Toc504319379)

[Krav & Leverans 4](#_Toc504319380)

[Apendex 4](#_Toc504319381)

[UML 4](#_Toc504319382)

[Syntax 4](#_Toc504319383)

[Controller 4](#_Toc504319384)

[DBConnector.java 4](#_Toc504319385)

[RandGenerator.java 5](#_Toc504319386)

[Model 9](#_Toc504319387)

[Adventure.java 9](#_Toc504319388)

[Room.java 13](#_Toc504319389)

[Character 18](#_Toc504319390)

[Character.java 18](#_Toc504319391)

[CharacterInterface.java 24](#_Toc504319392)

[CollectionOfCharacters 24](#_Toc504319393)

[------ 24](#_Toc504319394)

[Knight.java 25](#_Toc504319395)

[Thief.java 26](#_Toc504319396)

[Wizard.java 27](#_Toc504319397)

[Monster 29](#_Toc504319398)

[Monter.java 29](#_Toc504319399)

[GiantSpider.java 31](#_Toc504319400)

[Orc.java 31](#_Toc504319401)

[Skeleton.java 31](#_Toc504319402)

[Troll.java 32](#_Toc504319403)

[View 32](#_Toc504319404)

[Combat.java 32](#_Toc504319405)

[Menu.java 35](#_Toc504319406)

[Text.java 48](#_Toc504319407)

[JsonUtil.java 51](#_Toc504319408)

[Main.java 52](#_Toc504319409)

# Introdution

Dungeon Run är ett textbaserat äventyrsspel för en spelare. Det spelas genom att göra val i menyer som innehåller olika alternativ. Man väljer vilken typ av hjälte man vill spela, för att sedan utforska en karta med slumpmässigt innehåll i jakt på skatter. Men se upp för monster! Det gäller att samla på sig så mycket skatter som möjligt och att hitta ut med livet i behåll.

Detta är ett speldesigndokument på hög nivå som beskriver vad spelet ska innehålla. Om något är oklart ska man fråga produktägaren Johan Svahn.Tropa de elit gamming managment.

Spelet är utvecklad av Devopsgrupen.

Kim, Burhan,Axel,Peter, Kassim

# Syfte (Behövs syfte??)

# Krav & Leverans

* Spelet ska fungera från början till slut
* Spelet ska uppfylla funktionerna enligt detta designdokument
* Spelet ska kunna köras på Windows  Spelet ska levereras kompilerat (som exe-fil)
* Spelet ska levereras med ett dokument som beskriver kod-arkitekturen på en hög nivå
* Om spelet innehåller grafiska element så är det en bonus .

Leverans till produktägaren ska ske torsdagen 25:e januari på plats i Nackademins lokaler, efter en genomförd demonstration.

# Apendex

## UML

## Syntax

### Controller

##### DBConnector.java

-----

**package** controller;  
  
**import** java.sql.Connection;  
**import** java.sql.DriverManager;  
  
  
**public class** DBConnector {  
  
 **public static void** main(String[] args) {  
 **try** {  
 Connection conn = *getConnection*();  
 System.***out***.println(conn);  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 **public static** Connection getConnection() **throws** Exception {  
 **try** {  
 String driver = (**"com.mysql.jbdc.Driver"**);  
 String url = **"jbdc:mysql//dbtropa.cohjnbb8mlur.eu-west-3.rds.amazonaws.com:3306/tropa"**;  
 String username = **"peter"**;  
 String password = **"tropa1337"**;  
 Class.*forName*(driver);  
 Connection connection = DriverManager.*getConnection*(url, username, password);  
 System.***out***.println(**"It worked"**);  
 **return** connection;  
 } **catch** (Exception e) {  
 System.***out***.println(e);  
 }  
 **return null**;  
 }  
  
}

-----

##### RandGenerator.java

-------

*// Fig. 6.7: RandomInt.java  
// Shifted, scaled random integers***package** controller;  
**import** model.Monster.\*;  
**import** model.Room;  
**import** model.Character.Character;  
  
**import** java.util.ArrayList;  
**import** java.util.Random;  
  
  
**public class** RandGenerator {  
  
  
 *//Deklaration av attributer.* **private static boolean** *additionalMonster* = **false**;  
 **private static boolean** *firstMonster* = **true**;  
 **private static int** *level3MonsterLimit* = 3;  
 **private static int** *counter* = 1;  
  
  
 **public** RandGenerator(){}  
  
 *//method som skapar / genererar Monsters* **public void** generateMonsters() {  
  
  
  
  
  
 }  
  
 *//Metod som återställer antall generation Monsterfor samtliga rum.* **public void** resetVariables(){  
 **this**.*additionalMonster* = **false**;  
 **this**.*firstMonster* = **true**;  
 }  
  
 *//En lista skapas på samtliga monster som kan generaras samt sanolikhet att den spawnas för varje nivå.* **public static** ArrayList<Monster> rollTheDice(**int** level){  
 ArrayList<Monster> listOfMonsters = **new** ArrayList<Monster>();  
  
 **boolean** spawnSpider = *spawnMonster*(level,20);  
  
 **boolean** spawnSkeleton = *spawnMonster*(level, 15);  
  
 **boolean** spawnOrc = *spawnMonster*(level, 10);  
  
 **boolean** spawnTroll = *spawnMonster*(level, 5);  
  
  
 **if**(spawnSpider){  
 Monster giantSpider = **new** GiantSpider();  
 listOfMonsters.add(giantSpider);  
 }  
 **if**(spawnSkeleton){  
 Monster skeleton = **new** Skeleton();  
 listOfMonsters.add(skeleton);  
 }  
 **if**(spawnOrc){  
 Monster orc = **new** Orc();  
 listOfMonsters.add(orc);  
 }  
 **if**(spawnTroll){  
 Monster troll = **new** Troll();  
 listOfMonsters.add(troll);  
 }  
  
 **return** listOfMonsters;  
  
 }  
  
 *//genererar/spawn slumpmässiga monster mellan 0 - 100 i boundery.* **private static boolean** spawnMonster(**int** level, **int** chance) {  
 **if** (*firstMonster* || *additionalMonster*) {  
 Random rand = **new** Random();  
 **int** randomInt = rand.nextInt(100);  
  
 *//Verifierar ifall om det redan finns ett på Level 1 om inte genererar det en monster enligt chance nivå* **if** (randomInt <= chance) {  
  
 **if** (level == 1) {  
 **if** (*firstMonster*) {  
 *firstMonster* = **false**;  
 **return true**;  
 }  
 **else**{**return false**; }  
 }  
  
 *//Verifierar ifall om det redan finns ett monster på Level 2 oavsätt so lägger den till ett till monster.* **if** (level == 2) {  
 **if** (*firstMonster*) {  
 *firstMonster* = **false**;  
 *additionalMonster* = **true**;  
  
 **return true**;  
 }  
  
 *//Verifierar ifall om det redan finns ett monster.  
 //om det fins så lägger den till ett till monster om randomInt är större än 50.* randomInt = rand.nextInt(100);  
  
 **if** (randomInt <= 50 && *additionalMonster* == **true**) {  
 *//spawna ett monster  
 additionalMonster* = **false**;  
 **return true**;  
 } **else if** (randomInt <= 50 && *additionalMonster* == **false**) {  
 *//spawna inte ett monster* **return false**;  
 }  
 }  
  
 *//Genererar Monster oavsätt om det redan finns eller ej.* **if** (level == 3) {  
 **if**(*counter* <= *level3MonsterLimit*){  
 *counter* += 1;  
 *//spawna monster* **return true**;  
 }  
 **else**{  
 randomInt = rand.nextInt(100);  
  
 **if** (randomInt <= 50) {  
 *//spawna ett monster* **return true**;  
 }  
 **else**{  
 **return false**;  
 }  
 }  
 }  
 }  
  
  
 }  
 **return false**;  
 }  
  
 *// Metod som genererar Items / Treasure.* **private int** ItemGenerator(Character character){  
 Random rand = **new** Random();  
 **int** randomInt = rand.nextInt(100);  
 **int** treasureTotalWorth = 0;  
  
 **try** {  
 **if** (randomInt <= 40) {  
 System.***out***.println(**"Du hittade lösa slantar "**);  
 character.raiseCoin();  
 Thread.*sleep*(500);  
 treasureTotalWorth += 2;  
 }  
  
 randomInt = rand.nextInt(100);  
  
 **if** (randomInt <= 20) {  
 System.***out***.println(**"Du hittade en pengapung "**);  
 character.raiseCoinBag();  
 Thread.*sleep*(500);  
 treasureTotalWorth += 6;  
 }  
  
 randomInt = rand.nextInt(100);  
  
 **if** (randomInt <= 15) {  
 System.***out***.println(**"Du hittade guldsmycken"**);  
 character.raiseGoldJewelry();  
 Thread.*sleep*(500);  
 treasureTotalWorth += 10;  
 }  
  
 randomInt = rand.nextInt(100);  
  
 **if** (randomInt <= 10) {  
 System.***out***.println(**"Du hittade en ädelsten"**);  
 character.raiseGemstone();  
 Thread.*sleep*(500);  
 treasureTotalWorth += 14;  
 }  
  
 randomInt = rand.nextInt(100);  
  
  
 **if** (randomInt <= 5) {  
 System.***out***.println(**"Du hittade en liten skattkista !"**);  
 character.raiseTreasureBox();  
 Thread.*sleep*(500);  
 treasureTotalWorth += 20;  
 }  
 }  
 **catch**(InterruptedException e){  
 e.printStackTrace();  
 }  
  
 **return** treasureTotalWorth;  
 }  
  
 **public static boolean** isAdditionalMonster() {  
 **return** *additionalMonster*;  
 }  
  
 **public static void** setAdditionalMonster(**boolean** additionalMonster) {  
 RandGenerator.*additionalMonster* = additionalMonster;  
 }  
  
 **public static boolean** isFirstMonster() {  
 **return** *firstMonster*;  
 }  
  
 **public static void** setFirstMonster(**boolean** firstMonster) {  
 RandGenerator.*firstMonster* = firstMonster;  
 }  
  
 **public static int** getLevel3MonsterLimit() {  
 **return** *level3MonsterLimit*;  
 }  
  
 **public static void** setLevel3MonsterLimit(**int** level3MonsterLimit) {  
 RandGenerator.*level3MonsterLimit* = level3MonsterLimit;  
 }  
  
 **public static int** getCounter() {  
 **return** *counter*;  
 }  
  
 **public static void** setCounter(**int** counter) {  
 RandGenerator.*counter* = counter;  
 }  
  
}

--------

### Model

##### Adventure.java

----------

**package** model;  
**import** model.Character.Character;  
**import** model.Monster.Monster;  
**import** view.Combat;  
**import** view.Menu;  
  
**import** java.io.IOException;  
**import** java.util.ArrayList;  
**import** java.util.Scanner;  
  
**public class** Adventure {  
  
 **private int level**;  
 **private** Character **myCharacter**;  
 **private int startPosition**;  
 **private int startPositionX**;  
 **private int startPositionY**;  
 **private** ArrayList<Monster> **monsterToFight**;  
 **private** Room **rooms**;  
 **private** Menu **menu**;  
 **private** Scanner **scanner**;  
  
  
 **public** Adventure(Menu menu) {  
 **this**.**menu**=menu;  
 **this**.**menu**.setMyAdventure(**this**);  
 **this**.**startPosition** = **this**.**menu**.getStartingCorner();  
 **this**.**level** = **this**.**menu**.getSubmitDifficulty();  
 **scanner** = **new** Scanner(System.***in***);  
 }  
  
 **public void** startNewRoom() **throws** IOException, InterruptedException {  
 getStartingPositions();  
 **this**.**rooms** = **new** Room(**startPositionX**,**startPositionY**,**level**);  
 gameRound();  
  
 }  
  
 **public void** gameRound() **throws** IOException, InterruptedException {  
  
 String w = **"W"**;  
 String a = **"A"**;  
 String d = **"D"**;  
 String s = **"S"**;  
  
 **while**(**true**){  
  
 **menu**.clearWindow();  
 **rooms**.printRoom();  
 System.***out***.println(**"Which direction do u want to go?\n\n"** +  
 **"[W]Go Up\n"** +  
 **"[A]Go Left\n"** +  
 **"[D]Go Right\n"** +  
 **"[S]Go Down\n\n\n"** +  
 **"\t\t\t\t[0]EXIT"**);  
  
 String gameRoundInput = **scanner**.nextLine().toUpperCase();  
  
  
 **switch** (gameRoundInput){  
  
 **case "W"**:  
 makeAMove(**"W"**);  
 **break**;  
 **case "A"**:  
 makeAMove(**"A"**);  
 **break**;  
 **case "D"**:  
 makeAMove(**"D"**);  
 **break**;  
 **case "S"**:  
 makeAMove(**"S"**);  
 **break**;  
  
 **case "0"**:  
 System.*exit*(9);  
 **break**;  
  
 }

}

}  
  
 **public void** flee(){  
 String[][] room;  
 room=**rooms**.getRooms();  
 room[**rooms**.getOldPlayerPositionX()][**rooms**.getOldPlayerPositionY()]=**rooms**.getPlayerPosition();  
 **rooms**.flee();  
  
 **rooms**.setRooms(room);  
 }  
  
 **public void** startFight() **throws** IOException, InterruptedException {  
  
 **this**.**monsterToFight**=**this**.**rooms**.getMonstersToFight();  
 Combat combat = **new** Combat();  
 **for** (**int** i = 0; i < **this**.**monsterToFight**.size(); i++){  
  
 combat.combatStart(**this**.**monsterToFight**.get(i),**this**.**menu**.getMyCharacter(),**this**.**rooms**);  
  
 }  
  
 }  
  
 **public void** makeAMove(String direction) **throws** IOException, InterruptedException {  
 Boolean fight=**false**;

**if**(direction.equals(**"W"**)){  
 fight=**this**.**rooms**.moveUp();  
  
 **if**(fight){  
 startFight();  
 }  
  
 }

**if**(direction.equals(**"S"**)){  
  
 fight=**this**.**rooms**.moveDown();  
  
 **if**(fight){  
 startFight();  
 }  
 }  
  
 **if**(direction.equals(**"A"**)){  
  
 fight=**this**.**rooms**.moveLeft();  
  
 **if**(fight){  
 startFight();  
 }

}  
 **if**(direction.equals(**"D"**)){  
  
 fight=**this**.**rooms**.moveRight();  
  
 **if**(fight){  
 startFight();  
 }  
 }  
 }  
  
 **public** Character getMyCharacter() {  
 **return myCharacter**;  
 }  
  
 **public void** setMyCharacter(Character myCharacter) {  
 **this**.**myCharacter** =**new** Character();  
 **this**.**myCharacter** = myCharacter;  
 }  
  
 **public** Room getRooms() {  
 **return rooms**;  
 }  
  
 */\*This function sets the starting position of the character based on what level(1,2,3) you choose and which startPosition(1,2,3,4) you selected  
 Example: if constructor gets 1,2 you will get this layout of rooms and you will spawn where the 2 is located  
 1XX2  
 XXXX  
 3XX4  
  
 \*/* **public void** getStartingPositions(){  
  
 *//level 1= 4x4 rooms* **if**(**this**.**level**==1) {  
 **if** (**this**.**startPosition** == 1) {  
 **this**.**startPositionX** = 0;  
 **this**.**startPositionY** = 0;  
  
  
 }  
 **if** (**this**.**startPosition** == 2) {  
 **this**.**startPositionX** = 0;  
 **this**.**startPositionY** = 3;  
  
 }  
 **if** (**this**.**startPosition** == 3) {  
 **this**.**startPositionX** = 3;  
 **this**.**startPositionY** = 0;  
  
 }  
 **if** (**this**.**startPosition** == 4) {  
 **this**.**startPositionX** = 3;  
 **this**.**startPositionY** = 3;  
  
 }  
 }  
 *//level 2= 5x5 rooms* **if**(**this**.**level**==2){  
 **if**(**this**.**startPosition**==1){  
 **this**.**startPositionX**=0;  
 **this**.**startPositionY**=0;  
  
  
 }  
 **if**(**this**.**startPosition**==2){  
 **this**.**startPositionX**=0;  
 **this**.**startPositionY**=4;  
  
 }  
 **if**(**this**.**startPosition**==3){  
 **this**.**startPositionX**=4;  
 **this**.**startPositionY**=0;  
  
 }  
 **if**(**this**.**startPosition**==4){  
 **this**.**startPositionX**=4;  
 **this**.**startPositionY**=4;  
  
 }  
 }  
  
 *//level 3= 8x8 rooms* **if**(**this**.**level**==3){  
 **if**(**this**.**startPosition**==1){  
 **this**.**startPositionX**=0;  
 **this**.**startPositionY**=0;  
  
  
 }  
 **if**(**this**.**startPosition**==2){  
 **this**.**startPositionX**=0;  
 **this**.**startPositionY**=7;  
  
 }  
 **if**(**this**.**startPosition**==3){  
 **this**.**startPositionX**=7;  
 **this**.**startPositionY**=0;  
  
 }  
 **if**(**this**.**startPosition**==4){  
 **this**.**startPositionX**=7;  
 **this**.**startPositionY**=7;  
  
 }  
 }  
  
 }  
  
}

--------

##### Room.java

-------

**package** model;  
  
**import** model.Monster.Monster;  
  
**import** java.util.ArrayList;  
  
**public class** Room {  
 **private int playerPositionX**;  
 **private int playerPositionY**;  
 **private** String **playerPosition**=**"[P]"**;  
 **private int level**;  
 **private** String[][] **rooms**;  
 **private** ArrayList<Monster> **monstersToFight**;  
 **private int oldPlayerPositionX**;  
 **private int oldPlayerPositionY**;  
  
  
 **public** Room(**int** playerPositionX, **int** playerPositionY, **int** level) {  
 **this**.**playerPositionX** = playerPositionX;  
 **this**.**playerPositionY** = playerPositionY;  
 **this**.**level** = level;  
  
 createGame();  
 fillRoom();  
 }  
  
 **public int** getLevel() {  
 **return level**;  
 }  
  
 **public void** printRoom(){  
 String stringToReturn=**""**;  
  
 **for**(**int** rows=0;rows<**this**.**rooms**.**length**;rows++){  
 stringToReturn+=rows;  
 **for**(**int** columns=0;columns <**this**.**rooms**[rows].**length**;columns++){  
 stringToReturn+=columns;  
  
 System.***out***.print(**rooms**[rows][columns] );}  
 System.***out***.println();}  
  
 }  
*//* **public void** flee(){  
 **this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**]=**"[X]"**;  
 **this**.**rooms**[**oldPlayerPositionX**][**oldPlayerPositionY**]=**playerPosition**;  
 **this**.**playerPositionX**=**oldPlayerPositionX**;  
 **this**.**playerPositionY**=**oldPlayerPositionY**;  
 }  
  
  
  
 **public boolean** moveDown(){  
 **this**.**oldPlayerPositionX**=**this**.**playerPositionX**;  
 **this**.**oldPlayerPositionY**=**this**.**playerPositionY**;  
 String Roomcondition;  
 **try**{  
  
 **this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**] =**"[ ]"**;  
 **this**.**playerPositionX**++;  
 Roomcondition=**this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**];  
 **this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**]=**playerPosition**;  
  
 **if** (Roomcondition.equals(**"[X]"**)){  
 **this**.**monstersToFight**=controller.RandGenerator.*rollTheDice*(**this**.**level**);  
  
 **return true**;  
  
 }**else**{  
 System.***out***.println(**"You have been here before"**);  
 **return false**;  
 }  
  
  
 }**catch** (ArrayIndexOutOfBoundsException e){  
 **this**.**playerPositionX**=**oldPlayerPositionX**;  
 **this**.**playerPositionY**=**oldPlayerPositionY**;  
 System.***out***.println(**"You can't walk that way"**);  
 **this**.**rooms**[**oldPlayerPositionX**][**oldPlayerPositionY**]=**playerPosition**;  
 **return false**;  
  
  
 }  
 }  
  
 **public boolean** moveUp(){  
 **this**.**oldPlayerPositionX**=**this**.**playerPositionX**;  
 **this**.**oldPlayerPositionY**=**this**.**playerPositionY**;  
 String Roomcondition;  
 **try**{  
  
 **this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**] =**"[ ]"**;  
 **this**.**playerPositionX**--;  
 Roomcondition=**this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**];  
 **this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**]=**playerPosition**;  
  
  
 **if** (Roomcondition.equals(**"[X]"**)){  
 **this**.**monstersToFight**=controller.RandGenerator.*rollTheDice*(**this**.**level**);  
 **return true**;  
  
 }**else**{  
 System.***out***.println(**"You have been here before"**);  
 **return false**;  
 }  
  
 }**catch** (ArrayIndexOutOfBoundsException e){  
 **this**.**playerPositionX**=**oldPlayerPositionX**;  
 **this**.**playerPositionY**=**oldPlayerPositionY**;  
 System.***out***.println(**"You can't walk that way"**);  
 **this**.**rooms**[**oldPlayerPositionX**][**oldPlayerPositionY**]=**playerPosition**;  
 **return false**;  
  
 }  
  
  
  
 }  
  
 **public boolean** moveLeft(){  
 **this**.**oldPlayerPositionX**=**this**.**playerPositionX**;  
 **this**.**oldPlayerPositionY**=**this**.**playerPositionY**;  
 String Roomcondition;  
 **try**{  
  
 **this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**] =**"[ ]"**;  
 **this**.**playerPositionY**--;  
 Roomcondition=**this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**];  
 **this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**]=**playerPosition**;  
  
 **if** (Roomcondition.equals(**"[X]"**)){  
 **this**.**monstersToFight**=controller.RandGenerator.*rollTheDice*(**this**.**level**);  
 **return true**;  
  
 }**else**{  
 System.***out***.println(**"You have been here before"**);  
 **return false**;  
 }  
  
 }**catch** (ArrayIndexOutOfBoundsException e){  
 **this**.**playerPositionX**=**oldPlayerPositionX**;  
 **this**.**playerPositionY**=**oldPlayerPositionY**;  
 System.***out***.println(**"You can't walk that way"**);  
 **this**.**rooms**[**oldPlayerPositionX**][**oldPlayerPositionY**]=**playerPosition**;  
 **return false**;  
  
 }  
  
 }  
  
 **public boolean** moveRight(){  
 **this**.**oldPlayerPositionX**=**this**.**playerPositionX**;  
 **this**.**oldPlayerPositionY**=**this**.**playerPositionY**;  
 String Roomcondition;  
 **try**{  
  
 **this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**] =**"[ ]"**;  
 **this**.**playerPositionY**++;  
 Roomcondition=**this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**];  
 **this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**]=**playerPosition**;  
  
 **if** (Roomcondition.equals(**"[X]"**)){  
 **this**.**monstersToFight**=controller.RandGenerator.*rollTheDice*(**this**.**level**);  
 **return true**;  
  
 }**else**{  
 System.***out***.println(**"You have been here before"**);  
 **return false**;  
 }  
  
 }**catch** (ArrayIndexOutOfBoundsException e){  
 **this**.**playerPositionX**=**oldPlayerPositionX**;  
 **this**.**playerPositionY**=**oldPlayerPositionY**;  
 System.***out***.println(**"You can't walk that way"**);  
 **this**.**rooms**[**oldPlayerPositionX**][**oldPlayerPositionY**]=**playerPosition**;  
 **return false**;  
  
 }  
  
 }  
  
  
 **public void** fillRoom(){  
  
 **for**(**int** rows=0;rows<**this**.**rooms**.**length**;rows++) {  
 **for** (**int** columns = 0; columns < **this**.**rooms**[rows].**length**; columns++) {  
 **this**.**rooms**[rows][columns] = **"[X]"**;  
 *//System.out.print(rooms[rows][columns] + "\t");* }  
  
 }  
  
 **this**.**rooms**[**this**.**playerPositionX**][**this**.**playerPositionY**]=**playerPosition**;  
  
  
  
 }  
  
 */\*  
 \*This function creates rooms based of level and sets player position to one of the rooms  
 \*/* **public void** createGame(){  
  
 **if**(**level**==1){  
 **this**.**rooms** = **new** String[4][4];  
  
  
 }  
  
 **if**(**level**==2){  
 **this**.**rooms** = **new** String[5][5];  
  
  
 }  
  
 **if**(**level**==3){  
 **this**.**rooms** = **new** String[8][8];  
  
  
  
  
 }  
  
  
  
 }  
  
 **public** ArrayList<Monster> getMonstersToFight() {  
 **return monstersToFight**;  
 }  
  
 **public** String[][] getRooms() {  
 **return rooms**;  
 }  
  
 **public** String getPlayerPosition() {  
 **return playerPosition**;  
 }  
  
 **public int** getOldPlayerPositionX() {  
 **return oldPlayerPositionX**;  
 }  
  
 **public int** getOldPlayerPositionY() {  
 **return oldPlayerPositionY**;  
 }  
  
 **public void** setRooms(String[][] rooms) {  
 **this**.**rooms** = rooms;  
 }  
}

---------

#### Character

##### Character.java

-------

**package** model.Character;  
  
**import** model.Monster.Monster;  
  
**import** java.io.Serializable;  
**import** java.util.Objects;  
**import** java.util.Random;  
**import** java.util.concurrent.ThreadLocalRandom;  
  
**public class** Character **implements** Serializable,CharacterInterface{  
 **private** String **classType**;  
 **private** String **name**;  
 **private int initiative**;  
 **private int hp**;  
 **private int attack**;  
 **private int agility**;  
 **private int coin**;  
 **private int coinBag**;  
 **private int goldJewelry**;  
 **private int gemstone**;  
 **private int treasureBox**;  
 **private boolean firstHit** = **true**;  
 **private int giantSpidersSlain**;  
 **private int skeletonsSlain**;  
 **private int orcSlain**;  
 **private int trollSlain**;  
  
 *//Character constructor and all get and setters.* **public** Character(String classType,String name, **int** initiative, **int** hp, **int** attack, **int** agility, **int** coin, **int** coinBag, **int** goldJewelry, **int** gemstone, **int** treasureBox) {  
 **this**.**classType** = classType;  
 **this**.**name** =name;  
 **this**.**initiative** = initiative;  
 **this**.**hp** = hp;  
 **this**.**attack** = attack;  
 **this**.**agility** = agility;  
 **this**.**coin** = coin;  
 **this**.**coinBag** = coinBag;  
 **this**.**goldJewelry** = goldJewelry;  
 **this**.**gemstone** = gemstone;  
 **this**.**treasureBox** = treasureBox;  
  
 }  
  
 **public** Character() {  
 }  
  
 *// used from view, searches and loads saved character* **public** Character loadCharacterFromStorage(String name){  
  
 Character myCharacter;  
  
 *// search storage for match and return the right object with given name* **return null**;  
 }  
  
 **public void** attackMonster(Monster monster){  
 **int** attackDamage = 0;  
 **int** monsterAgility = 0;  
  
 *//Rullar varje tärning och adderar det till attackdamage variablen* **for** (**int** i = 1; i <= **this**.**attack**; i++){  
 attackDamage = ThreadLocalRandom.*current*().nextInt(1, 6 + 1) + attackDamage;  
 }  
  
 *//variabel för att spara monstrets agilitiy roll* **for** (**int** i = 1; i <= monster.getAgility(); i++){  
 monsterAgility = ThreadLocalRandom.*current*().nextInt(1, 6 + 1) + monsterAgility ;  
 }  
  
 *//bara development syfte för att se rollsen* System.***out***.println(**"Your attack: "** + attackDamage);  
 System.***out***.println(**"Monster agility: "** + monsterAgility);  
 System.***out***.println(**" "**);  
  
 *//kollar om spelarens attack är högre än monstrets agility, applicerar thiefs passive om spelaren är en* **if** (attackDamage>monsterAgility){  
  
 **if**(**this**.**classType**.equals(**"Thief"**)){  
 **int** critChance = ThreadLocalRandom.*current*().nextInt(1,4 + 1);  
 **if**(critChance == 1){  
 System.***out***.println(**"You crit the "** + monster.getClassType());  
 monster.setHp(monster.getHp()- 2);  
 }  
 **else**{  
 System.***out***.println(**"You attack the "** + monster.getClassType());  
 monster.takeDamage();  
 }  
 }  
 **else** {  
  
 System.***out***.println(**"You attack the "** + monster.getClassType());  
 monster.takeDamage();  
 }  
 }  
 **else if** (attackDamage < monsterAgility){  
 System.***out***.println(**"Your attack missed! "**);  
 }  
 }  
 *//Tar in ett monster objekt och använder dens attribut och funktioner* **public void** defendAttack(Monster monster){  
  
 *//Variabler för att komma ihåg spelarens smidighets roll och monstrets attack roll* **int** monsterAttackDamage = 0;  
 **int** playerAgility = 0;  
  
  
 *//rolls för monstrets skada och spelarens agilitiy* **for** (**int** i = 1; i <=monster.getAttack(); i++){  
 monsterAttackDamage = ThreadLocalRandom.*current*().nextInt(1, 6 + 1) + monsterAttackDamage;  
 }  
  
  
 **for** (**int** i = 1; i <= **this**.**agility**; i++){  
 playerAgility = ThreadLocalRandom.*current*().nextInt(1, 6 + 1) + playerAgility;  
 }  
 System.***out***.println(**" "**);  
  
 *//Kollar om monstrets attack träffar eller inte, om den träffar kollar den om spelaren är en knight och om detta är första slaget* **if**(monsterAttackDamage<playerAgility){  
 System.***out***.println(monster.getClassType() + **"s attack missed!"** + **"\n"**);  
 }  
 **else if**(monsterAttackDamage>playerAgility){  
 **if**(**firstHit** && **this**.**classType**.equals(**"Knight"**)){  
 System.***out***.println(**"You block "** + monster.getClassType() + **" attack"**);  
 **firstHit** = **false**;  
 }  
 **else** {  
 System.***out***.println(monster.getClassType() + **"s hit!"**);  
 **this**.**hp** = **this**.**hp** - 1;  
 }  
 }  
 }  
 *//Returnar true om spelaren kan fly och false om inte* **public boolean** flee(){  
 Random rand = **new** Random();  
 *//wizards passiva ger 80% till att fly* **if** (**this**.**classType**.equals(**"Wizard"**)){  
 **if** (rand.nextInt(100)<80){  
 **return true**;  
 }  
 **else**{  
 **return false**;  
 }  
 }  
 **else** {  
 **if** (rand.nextInt(100) < (**this**.**agility** \* 10)) {  
 **return true**;  
 } **else** {  
 **return false**;  
 }  
 }  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Character{"** +  
 **"classType='"** + **classType** + **'\''** +  
 **", name='"** + **name** + **'\''** +  
 **", initiative="** + **initiative** +  
 **", hp="** + **hp** +  
 **", attack="** + **attack** +  
 **", agility="** + **agility** +  
 **", coin="** + **coin** +  
 **", coinBag="** + **coinBag** +  
 **", goldJewelry="** + **goldJewelry** +  
 **", gemstone="** + **gemstone** +  
 **", treasureBox="** + **treasureBox** +  
 **", firstHit="** + **firstHit** +  
 **'}'**;  
 }  
  
  
 @Override  
 **public** String previewLogo() {  
 **return null**;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public void** setCoin(**int** coin) {  
 **this**.**coin** = coin;  
 }  
  
 **public void** setCoinBag(**int** coinBag) {  
 **this**.**coinBag** = coinBag;  
 }  
  
 **public void** setGoldJewelry(**int** goldJewelry) {  
 **this**.**goldJewelry** = goldJewelry;  
 }  
  
 **public void** setGemstone(**int** gemstone) {  
 **this**.**gemstone** = gemstone;  
 }  
  
 **public void** setTreasureBox(**int** treasureBox) {  
 **this**.**treasureBox** = treasureBox;  
 }  
  
 **public int** getCoin() {  
 **return coin**;  
 }  
 **public void** raiseSpidersSlain() {  
 **this**.**giantSpidersSlain**+=1;  
 }  
 **public void** raiseSkeletonsSlain(){  
 **this**.**skeletonsSlain**+=1;  
 }  
  
 **public void** raiseCoin() {  
 **this**.**coin**+=1;  
 }  
  
 **public int** getCoinBag() {  
 **return coinBag**;  
 }  
  
 **public void** raiseCoinBag() {  
 **this**.**coinBag**+=1;  
 }  
  
 **public int** getGoldJewelry() {  
 **return goldJewelry**;  
 }  
  
 **public void** raiseGoldJewelry() {  
 **this**.**goldJewelry**+=1;  
 }  
  
 **public int** getGemstone() {  
 **return gemstone**;  
 }  
  
 **public void** raiseGemstone() {  
 **this**.**gemstone**+=1;  
 }  
  
 **public int** getTreasureBox() {  
 **return treasureBox**;  
 }  
  
 **public void** raiseTreasureBox() {  
 **this**.**treasureBox**+=1;  
 }  
  
 **public int** getInitiative() {  
 **return initiative**;  
 }  
  
 **public void** setInitiative(**int** initiative) {  
 **this**.**initiative** = initiative;  
 }  
  
 **public int** getHp() {  
 **return hp**;  
 }  
  
 **public void** setHp(**int** hp) {  
 **this**.**hp** = hp;  
 }  
  
 **public int** getAttack() {  
 **return attack**;  
 }  
  
 **public void** setAttack(**int** attack) {  
 **this**.**attack** = attack;  
 }  
  
 **public int** getAgility() {  
 **return agility**;  
 }  
  
 **public void** setAgility(**int** agility) {  
 **this**.**agility** = agility;  
 }  
  
 **public** String getClassType() {  
 **return classType**;  
 }  
  
 *//Reglera om det ska vara det första slaget i en strid* **public void** setFirstHit(**boolean** trueFalse){  
 **this**.**firstHit** = trueFalse;  
 }  
  
 **public void** setClassType(String classType) {  
 **this**.**classType** = classType;  
 }  
  
 **public boolean** isFirstHit() {  
 **return firstHit**;  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (**this** == o) **return true**;  
 **if** (!(o **instanceof** Character)) **return false**;  
 Character character = (Character) o;  
 **return** getInitiative() == character.getInitiative() &&  
 getHp() == character.getHp() &&  
 getAttack() == character.getAttack() &&  
 getAgility() == character.getAgility() &&  
 getCoin() == character.getCoin() &&  
 getCoinBag() == character.getCoinBag() &&  
 getGoldJewelry() == character.getGoldJewelry() &&  
 getGemstone() == character.getGemstone() &&  
 getTreasureBox() == character.getTreasureBox() &&  
 isFirstHit() == character.isFirstHit() &&  
 Objects.*equals*(getClassType(), character.getClassType()) &&  
 Objects.*equals*(getName(), character.getName());  
 }  
  
 @Override  
 **public int** hashCode() {  
  
 **return** Objects.*hash*(getClassType(), getName(), getInitiative(), getHp(), getAttack(), getAgility(), getCoin(), getCoinBag(), getGoldJewelry(), getGemstone(), getTreasureBox(), isFirstHit());  
 }  
}

-------------

##### CharacterInterface.java

------

**package** model.Character;  
  
**public interface** CharacterInterface {  
  
 String previewLogo();  
}

-------

##### CollectionOfCharacters

##### ------

**package** model.Character;  
  
**import** java.io.Serializable;  
  
**public class** Wizard **extends** Character **implements** Serializable {  
 **public** Wizard() {  
 **super**(**"Wizard"**,**""**,6,4,9,4,0,0,0,0,0);  
*// this.setClassType("Wizard");  
// this.setName("");  
// this.setInitiative(6);  
// this.setHp(4);  
// this.setAttack(9);  
// this.setAgility(4);  
// this.setCoin(0);  
// this.setCoinBag(0);  
// this.setGoldJewelry(0);  
// this.setGemstone(0);  
// this.setTreasureBox(0);* }  
 *//This constructor is used when we load in a character from save file.* **public** Wizard(String classType, String name, **int** initiative, **int** hp, **int** attack, **int** agility, **int** coin, **int** coinBag, **int** goldJewelry, **int** gemstone, **int** treasureBox) {  
 **super**(classType, name, initiative, hp, attack, agility, coin, coinBag, goldJewelry, gemstone, treasureBox);  
 }  
  
 *//The Wizard character special attack.* **public void** starLight(){  
 *//Make monsters blind and have a 80% chance to run away* }  
  
  
 *// UI representation of the class with logo and attributes* **public** String previewLogo(){  
 **return "NAME: "** + **this**.getName() +**"\n"**+  
 **"\t\t\t\t ,/ \*\n"** +  
 **"WIZARD:\t\t\t\t \_,'/\_ |\n"** +  
 **"\t\t\t\t `(\")' ,'/\n"** +  
 **"\t\t\t\t \_ \_,-H-./ /\n"** +  
 **"Initiative= "** + **this**.getInitiative() + **"\t\t\t \\\_\\\_\\. /\n"** +  
 **"HP= "** + **this**.getHp() + **"\t\t\t\t )\" | (\n"** +  
 **"Attack= "** + **this**.getAttack() + **"\t\t\t \_\_/ H \\\_\_\n"** +  
 **"Agility= "** + **this**.getAgility() + **"\t\t\t \\ /|\\ /\n"** +  
 **"\t\t\t\t `--'|||`--'\n"** +  
 **"\t\t\t\t ==^==\n\n"** +  
 **"Special attack:\n"** +  
 **"Make monsters blind and have a 80% chance to run away.\n\n"** +  
 **"TreasureBox: "** + getTreasureBox() + **"\n"** +  
 **"Coin: "** + getCoin() + **"\tCoinBag: "** + getCoinBag() + **"\n"** +  
 **"GoldJewelry: "** + getGoldJewelry() + **"\tGemstone: "** + getGemstone();  
 }  
}

-------

##### Knight.java

------

**package** model.Character;  
  
**import** java.io.Serializable;  
  
**public class** Knight **extends** Character **implements** Serializable {  
 **public** Knight() {  
 **super**(**"Knight"**, **""**, 5, 9, 6, 4, 0, 0, 0, 0, 0);  
*// this.setClassType("Knight");  
// this.setName("");  
// this.setInitiative(5);  
// this.setHp(9);  
// this.setAttack(6);  
// this.setAgility(4);  
// this.setCoin(0);  
// this.setCoinBag(0);  
// this.setGoldJewelry(0);  
// this.setGemstone(0);  
// this.setTreasureBox(0);* }  
  
 *//This constructor is used when we load in a character from save file.* **public** Knight(String classType, String name, **int** initiative, **int** hp, **int** attack, **int** agility, **int** coin, **int** coinBag, **int** goldJewelry, **int** gemstone, **int** treasureBox) {  
 **super**(classType, name, initiative, hp, attack, agility, coin, coinBag, goldJewelry, gemstone, treasureBox);  
 }  
  
 *//The Knights character special attack.* **public void** ShieldBlock() {  
 *//Always blocks the first attack per fight* }  
  
 *// UI representation of the class with logo and attributes* **public** String previewLogo() {  
 **return "NAME: "** + **this**.getName() + **"\n"** +  
 **"\t\t\t\t /\n"** +  
 **"KNIGHT:\t\t\t\t ,~~ /\n"** +  
 **"\t\t\t\t \_ <=) \_/\_\n"** +  
 **"\t\t\t\t /I\\.=\"==.{>\n"** +  
 **"Initiative= "** + **this**.getInitiative() + **"\t\t\t \\I/-\\T/-'\n"** +  
 **"HP= "** + **this**.getHp() + **"\t\t\t\t /\_\\\n"** +  
 **"Attack= "** + **this**.getAttack() + **"\t\t\t // \\\\\_\n"** +  
 **"Agility= "** + **this**.getAgility() + **"\t\t\t \_I /\n\n"** +  
 **"Special attack:\n"** +  
 **"Always blocks the first attack per fight.\n\n"** +  
 **"TreasureBox: "** + getTreasureBox() + **"\n"** +  
 **"Coin: "** + getCoin() + **"\tCoinBag: "** + getCoinBag() + **"\n"** +  
 **"GoldJewelry: "** + getGoldJewelry() + **"\tGemstone: "** + getGemstone();  
 }  
  
  
}

--------

##### Thief.java

------

**package** model.Character;  
  
**import** java.io.Serializable;  
  
**public class** Thief **extends** Character **implements** Serializable {  
 **public** Thief() {  
 **super**(**"Thief"**,**""**,7,5,5,7,0,0,0,0,0);  
*// this.setClassType("Thief");  
// this.setName("");  
// this.setInitiative(7);  
// this.setHp(5);  
// this.setAttack(5);  
// this.setAgility(7);  
// this.setCoin(0);  
// this.setCoinBag(0);  
// this.setGoldJewelry(0);  
// this.setGemstone(0);  
// this.setTreasureBox(0);* }  
 *//This constructor is used when we load in a character from save file.* **public** Thief(String classType, String name, **int** initiative, **int** hp, **int** attack, **int** agility, **int** coin, **int** coinBag, **int** goldJewelry, **int** gemstone, **int** treasureBox) {  
 **super**(classType, name, initiative, hp, attack, agility, coin, coinBag, goldJewelry, gemstone, treasureBox);  
 }  
  
 *//The Thief's character special attack.* **public int** criticalHit(){  
 **int** attack = getAttack();  
 **return** attack\*2;  
 *//25% chance to make double damage every attack* }  
  
 *// UI representation of the class with logo and attributes* **public** String previewLogo(){  
 **return "NAME: "** + **this**.getName() +**"\n"**+  
 **"\t\t\t\t ,/|\\\n"** +  
 **"THIEF:\t\t\t\t //&')\n"** +  
 **"\t\t\t\t '')(\n"** +  
 **"\t\t\t\t (( )\n"** +  
 **"Initiative= "** + **this**.getInitiative() + **"\t\t\t )( (\n"** +  
 **"HP= "** + **this**.getHp() + **"\t\t\t\t (=M=[)####>\n"** +  
 **"Attack= "** + **this**.getAttack() + **"\t\t\t (( )\n"** +  
 **"Agility= "** + **this**.getAgility() + **"\t\t\t (( )\_\n"** +  
 **"\t\t\t\t ((\_\_,)\n\n"** +  
 **"Special attack:\n"** +  
 **"25% chance to make double damage every attack.\n\n"** +  
 **"TreasureBox: "** + getTreasureBox() + **"\n"** +  
 **"Coin: "** + getCoin() + **"\tCoinBag: "** + getCoinBag() + **"\n"** +  
 **"GoldJewelry: "** + getGoldJewelry() + **"\tGemstone: "** + getGemstone();  
  
 }  
  
  
}

---------

##### Wizard.java

------

**package** model.Character;  
  
**import** java.io.Serializable;  
  
**public class** Wizard **extends** Character **implements** Serializable {  
 **public** Wizard() {  
 **super**(**"Wizard"**,**""**,6,4,9,4,0,0,0,0,0);  
*// this.setClassType("Wizard");  
// this.setName("");  
// this.setInitiative(6);  
// this.setHp(4);  
// this.setAttack(9);  
// this.setAgility(4);  
// this.setCoin(0);  
// this.setCoinBag(0);  
// this.setGoldJewelry(0);  
// this.setGemstone(0);  
// this.setTreasureBox(0);* }  
 *//This constructor is used when we load in a character from save file.* **public** Wizard(String classType, String name, **int** initiative, **int** hp, **int** attack, **int** agility, **int** coin, **int** coinBag, **int** goldJewelry, **int** gemstone, **int** treasureBox) {  
 **super**(classType, name, initiative, hp, attack, agility, coin, coinBag, goldJewelry, gemstone, treasureBox);  
 }  
  
 *//The Wizard character special attack.* **public void** starLight(){  
 *//Make monsters blind and have a 80% chance to run away* }  
  
  
 *// UI representation of the class with logo and attributes* **public** String previewLogo(){  
 **return "NAME: "** + **this**.getName() +**"\n"**+  
 **"\t\t\t\t ,/ \*\n"** +  
 **"WIZARD:\t\t\t\t \_,'/\_ |\n"** +  
 **"\t\t\t\t `(\")' ,'/\n"** +  
 **"\t\t\t\t \_ \_,-H-./ /\n"** +  
 **"Initiative= "** + **this**.getInitiative() + **"\t\t\t \\\_\\\_\\. /\n"** +  
 **"HP= "** + **this**.getHp() + **"\t\t\t\t )\" | (\n"** +  
 **"Attack= "** + **this**.getAttack() + **"\t\t\t \_\_/ H \\\_\_\n"** +  
 **"Agility= "** + **this**.getAgility() + **"\t\t\t \\ /|\\ /\n"** +  
 **"\t\t\t\t `--'|||`--'\n"** +  
 **"\t\t\t\t ==^==\n\n"** +  
 **"Special attack:\n"** +  
 **"Make monsters blind and have a 80% chance to run away.\n\n"** +  
 **"TreasureBox: "** + getTreasureBox() + **"\n"** +  
 **"Coin: "** + getCoin() + **"\tCoinBag: "** + getCoinBag() + **"\n"** +  
 **"GoldJewelry: "** + getGoldJewelry() + **"\tGemstone: "** + getGemstone();  
 }  
}

-------

#### Monster

##### Monter.java

-----

**package** model.Monster;  
  
*// Declaration of main MonsterCharacter Attributes.***import** java.util.Objects;  
  
**public class** Monster {  
 **private** String **classType**;  
 **private int initiative**;  
 **private int hp**;  
 **private int attack**;  
 **private int agility**;  
 **private float commonality**;  
  
  
 **public** Monster(String classType, **int** initiative, **int** hp, **int** attack, **int** agility, **float** commonality) {  
 **this**.**classType** = classType;  
 **this**.**initiative** = initiative;  
 **this**.**hp** = hp;  
 **this**.**attack** = attack;  
 **this**.**agility** = agility;  
 **this**.**commonality** = commonality;  
  
 *//Calls Main Monster class method.* }  
 **public** Monster() {  
 }  
  
 *////* **public void** setClassType(String classType) {  
 **this**.**classType** = classType;  
 }  
  
 **public void** setInitiative(**int** initiative) {  
 **this**.**initiative** = initiative;  
 }  
  
 **public void** setHp(**int** hp) {  
 **this**.**hp** = hp;  
 }  
  
 **public void** setAttack(**int** attack) {  
 **this**.**attack** = attack;  
 }  
  
 **public void** setAgility(**int** agility) {  
 **this**.**agility** = agility;  
 }  
  
 **public void** setCommonality(**float** commonality) {  
 **this**.**commonality** = commonality;  
 }  
  
 **public** String getClassType() {  
 **return classType**;  
 }  
  
 **public int** getInitiative() {  
 **return initiative**;  
 }  
  
 **public int** getHp() {  
 **return hp**;  
 }  
  
 **public int** getAttack() {  
 **return attack**;  
 }  
  
 **public int** getAgility() {  
 **return agility**;  
 }  
  
 **public float** getCommonality() {  
 **return commonality**;  
 }  
  
 **public void** takeDamage(){  
 **this**.**hp** = **this**.**hp** - 1;  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (**this** == o) **return true**;  
 **if** (!(o **instanceof** Monster)) **return false**;  
 Monster monster = (Monster) o;  
 **return** getInitiative() == monster.getInitiative() &&  
 getHp() == monster.getHp() &&  
 getAttack() == monster.getAttack() &&  
 getAgility() == monster.getAgility() &&  
 Float.*compare*(monster.getCommonality(), getCommonality()) == 0 &&  
 Objects.*equals*(getClassType(), monster.getClassType());  
 }  
  
 @Override  
 **public int** hashCode() {  
  
 **return** Objects.*hash*(getClassType(), getInitiative(), getHp(), getAttack(), getAgility(), getCommonality());  
 }  
  
 **public** String toString() {  
 **return "Monster{"** +  
 **"classType='"** + **classType** + **'\''** +  
 **", initiative="** + **initiative** +  
 **", hp="** + **hp** +  
 **", attack="** + **attack** +  
 **", agility="** + **agility** +  
 **", commonality="** + **commonality** +  
 **'}'**;  
 }  
}

------

##### GiantSpider.java

----

**package** model.Monster;  
  
*// inheritance of main MonsterCharacter Attributes For GiantSpider.***public class** GiantSpider **extends** Monster {  
 **public** GiantSpider(){  
 **super**();  
 **this**.setClassType(**"GiantSpider"**);  
 **this**.setInitiative(7);  
 **this**.setHp(1);  
 **this**.setAttack(2);  
 **this**.setAgility(3);  
 **this**.setCommonality(20);  
  
 }  
  
  
  
}

------

##### Orc.java

------

**package** model.Monster;  
  
*// inheritance of main MonsterCharacter Attributes For Orc.***public class** Orc **extends** Monster {  
 **public** Orc (){  
 **super**();  
 **this**.setClassType(**"Orc"**);  
 **this**.setInitiative(6);  
 **this**.setHp(3);  
 **this**.setAttack(4);  
 **this**.setAgility(4);  
 **this**.setCommonality(10);  
  
 }  
  
}

------

##### Skeleton.java

------

**package** model.Monster;  
  
*// inheritance of main MonsterCharacter Attributes For Skeleton.***public class** Skeleton **extends** Monster {  
 **public** Skeleton(){  
 **super**();  
 **this**.setClassType(**"Skeleton"**);  
 **this**.setInitiative(4);  
 **this**.setHp(2);  
 **this**.setAttack(3);  
 **this**.setAgility(3);  
 **this**.setCommonality(15);  
  
 }  
  
}

--------

##### Troll.java

-----

**package** model.Monster;  
  
*// inheritance of main MonsterCharacter Attributes for Troll.***public class** Troll **extends** Monster {  
 **public** Troll (){  
 **super**();  
 **this**.setClassType(**"Troll"**);  
 **this**.setInitiative(2);  
 **this**.setHp(4);  
 **this**.setAttack(7);  
 **this**.setAgility(2);  
 **this**.setCommonality(5);  
  
 }  
  
}

------

### View

##### Combat.java

----

**package** view;  
  
**import** model.Adventure;  
**import** model.Monster.Monster;  
**import** model.Character.Character;  
**import** model.Room;  
  
**import** java.io.IOException;  
**import** java.util.Objects;  
**import** java.util.Scanner;  
**import** java.util.concurrent.ThreadLocalRandom;  
  
**public class** Combat {  
  
 **private int currentTurn**;  
 **private int counter** = 0;  
  
 **public void** combatStart(Monster monster, Character character, Room rooms) **throws** IOException, InterruptedException {  
 **boolean** cont = **false**;  
 **boolean** loop = **true**;  
 Scanner scan = **new** Scanner(System.***in***);  
 **int** playerInitiativeRoll = 0;  
 **int** monsterInitiativeRoll = 0;  
 clearWindow();  
 rooms.printRoom();  
  
 System.***out***.println(**"a "** + monster.getClassType() + **" has appeared!"**);  
  
  
 **while**(loop) {  
 System.***out***.println(**"Do you want to (A)ttack it or try to (F)lee ? "**);  
 String choice = scan.nextLine().toUpperCase();  
 **if** (choice.equals(**"A"**)) {  
 cont = **true**;  
 loop = **false**;  
 } **else if** (choice.equals(**"F"**)) {  
 **boolean** tryToFlee;  
 tryToFlee = character.flee();  
 **if** (tryToFlee) {  
 System.***out***.println(**"You fled from "** + monster.getClassType());  
 rooms.flee();  
 cont = **false**;  
 loop = **false**;  
 } **else** {  
 System.***out***.println(**"You've failed your escape "**);  
 cont = **true**;  
 loop = **false**;  
 }  
 }  
 **else** {  
 System.***out***.println(**"You need to choose between (A)ttack or (F)lee "**);  
 }  
 }  
 **if** (cont) {  
 character.setFirstHit(**true**);  
  
 *//rollar initiative för att se om monstret eller spelaren ska börja* **for** (**int** i = 1; i <= monster.getInitiative(); i++) {  
 monsterInitiativeRoll = ThreadLocalRandom.*current*().nextInt(1, 6 + 1) + monsterInitiativeRoll;  
 }  
  
 **for** (**int** i = 1; i <= character.getInitiative(); i++) {  
 playerInitiativeRoll = ThreadLocalRandom.*current*().nextInt(1, 6 + 1) + playerInitiativeRoll;  
 }  
 System.***out***.println(**"------------------------------"**);  
 System.***out***.println(**"You rolled... "** + playerInitiativeRoll);  
 System.***out***.println(monster.getClassType() + **" rolled... "** + monsterInitiativeRoll);  
 System.***out***.println(**"------------------------------"**);  
 System.***out***.println(**""**);  
 **try** {  
 Thread.*sleep*(2000);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 *//jämför rollsen och ger current turn till den som rollade högst (currentTurn 1 spelare, 0 monster)* **if** (playerInitiativeRoll > monsterInitiativeRoll) {  
 System.***out***.println(**"You go first! "**);  
 **this**.**currentTurn** = 1;  
 } **else** {  
 System.***out***.println(monster.getClassType() + **" goes first! "**);  
 **this**.**currentTurn** = 0;  
 }  
  
 *//Combat startas* **while** (**true**) {  
  
 **if** (**counter** == 2){  
 **try** {  
 Thread.*sleep*(2000);  
 **counter** = 0;  
 }  
 **catch**(InterruptedException e){  
 e.printStackTrace();  
 }  
  
 }  
 *//kollar om spelaren dog den förra rundan* **if** (character.getHp() <= 0) {  
 System.***out***.println(**"You died!"**);  
 *//sätt in våran death animation/funktion* **break**;  
 }  
  
 *//kollar om monstret dog förra rundan* **if** (monster.getHp() <= 0) {  
 System.***out***.println(monster.getClassType() + **" died!"**);  
 **try** {  
 Thread.*sleep*(2500);  
 }  
 **catch**(InterruptedException e){  
 e.printStackTrace();  
 }  
 **break**;  
 }  
  
 **if** (**this**.**currentTurn** == 1) {  
 *//frågar spelaren vad hen vill göra och spar det till en variabel* System.***out***.println(**"What do you want to do? "**);  
 System.***out***.println(**"(A)ttack, (F)lee"**);  
 String playerChoice = scan.nextLine().toUpperCase();  
  
 **if** (playerChoice.equals(**"A"**)) {  
 character.attackMonster(monster);  
 **this**.**currentTurn** = 0;  
 } **else if** (playerChoice.equals(**"F"**)) {  
  
 **boolean** flee = character.flee();  
  
 **if** (flee) {  
 System.***out***.println(**"You fled! "**);  
 rooms.flee();  
 **break**;  
 } **else** {  
 System.***out***.println(**"You've failed your escape. "**);  
 **this**.**currentTurn** = 0;  
 }  
  
  
 } **else** {  
 System.***out***.println(**"You need to (A)ttack or try to (F)lee"**);  
 }  
 } **else if** (**this**.**currentTurn** == 0) {  
 character.defendAttack(monster);  
 **this**.**currentTurn** = 1;  
 }  
 **this**.**counter**++;  
 }  
  
 }  
 }  
 **public void** clearWindow() **throws** IOException, InterruptedException {  
 **new** ProcessBuilder(**"cmd"**, **"/c"**, **"cls"**).inheritIO().start().waitFor();  
 }  
  
 **public int** getCurrentTurn() {  
 **return currentTurn**;  
 }  
  
 **public void** setCurrentTurn(**int** currentTurn) {  
 **this**.**currentTurn** = currentTurn;  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (**this** == o) **return true**;  
 **if** (!(o **instanceof** Combat)) **return false**;  
 Combat combat = (Combat) o;  
 **return** getCurrentTurn() == combat.getCurrentTurn();  
 }  
  
 @Override  
 **public int** hashCode() {  
  
 **return** Objects.*hash*(getCurrentTurn());  
 }  
}

------

##### Menu.java

------

**package** view;  
  
**import** model.Adventure;  
**import** model.Character.\*;  
**import** model.Character.Character;  
  
**import** java.io.IOException;  
**import** java.util.Scanner;  
  
**public class** Menu {  
  
 */\* Concept for terminal view were we use variables like booleans to control phases  
 \* and object references to try method involved in each phase route or functionality.  
 \* Inputs and outputs, while loops with switch cases for robust interaction for user  
 \* were we also consider the way movement gets impacted.  
 \* Graphical design concludes and will be developed as we go.\*/* **private int submitDifficulty** = 0;  
 **private int startingCorner** = 0;  
  
 **private boolean menuFirstPhase**;  
 **private boolean menuSecondPhase**;  
 **private boolean menuThirdPhase**;  
 **private** Scanner **scanner**;  
  
 *// Model attributes reserved for interaction by method/function calling* **private** Character **myCharacter**;  
 **private** Knight **myKnight**;  
 **private** Wizard **myWizard**;  
 **private** Thief **myThief**;  
 **private boolean validName**;  
  
 **private** CollectionOfCharacters **myCharacterData**;  
 **private** Adventure **myAdventure**;  
  
 **public** Menu() **throws** IOException, InterruptedException {  
 clearWindow();  
 **this**.**scanner** = **new** Scanner(System.***in***);  
 **this**.**menuFirstPhase** = **true**;  
 **this**.**menuSecondPhase** = **false**;  
 **this**.**myKnight** = **new** Knight();  
 **this**.**myWizard** = **new** Wizard();  
 **this**.**myThief** = **new** Thief();  
 **this**.**myCharacter** = **null**;  
 **this**.**myCharacterData** = **new** CollectionOfCharacters();  
 **myCharacterData**.openFunc(**"characterData"**);  
 **this**.**validName** = **false**;  
 **this**.**myAdventure** = **null**;  
  
 }  
  
 *// Graphical representation of classes with their attributes and values* **public void** classShowCase() {  
  
 System.***out***.println(**"\t\t\t["** + **myKnight**.getClassType() + **"]\t["** + **myWizard**.getClassType() + **"]\t["** + **myThief**.getClassType()  
 + **"]\n\nHealth:\t\t\t"** + **myKnight**.getHp() + **"\t\t\t"** + **myWizard**.getHp() + **"\t\t\t"** + **myThief**.getHp()  
 + **"\nAttack:\t\t\t"** + **myKnight**.getAttack() + **"\t\t\t"** + **myWizard**.getAttack() + **"\t\t\t"** + **myThief**.getAttack()  
 + **"\nAgility\t\t\t"** + **myKnight**.getAgility() + **"\t\t\t"** + **myWizard**.getAgility() + **"\t\t\t"** + **myThief**.getAgility()  
 + **"\nInitiative:\t\t"** + **myKnight**.getInitiative() + **"\t\t\t"** + **myWizard**.getInitiative() + **"\t\t\t"** + **myThief**.getInitiative()  
 + **"\n\n\n\t\t\t\t\t\t\t"** +  
 **"Enter to continue..."**);  
  
 **scanner**.nextLine();  
 }  
  
 *// UI initiation starts here* **public void** runMainMenu() **throws** IOException, InterruptedException {  
  
 clearWindow();  
 **while** (**menuFirstPhase**) {  
  
 System.***out***.println(**"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"** +  
 **"\n|\t\t\t\t\t\t|"** +  
 **"\n|\t\t\tMenu\t\t\t|"** +  
 **"\n|\t\t\t\t\t\t|"** +  
 **"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t[1]Create Character"** +  
 **"\n\t\t[2]Load Character"** +  
 **"\n\t\t[0]Exit"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t\t\t\t\t"**);  
  
  
 String inputForFirstPhase = **scanner**.nextLine().toUpperCase();  
  
 **switch** (inputForFirstPhase) {  
  
 *// Menu option 1 as follows* **case "1"**:  
 *// Create new character* **menuFirstPhase** = **false**;  
 **menuSecondPhase** = **true**;  
 clearWindow();  
 secondMenuCreation();  
 **break**;  
 **case "CREATE"**:  
 **menuFirstPhase** = **false**;  
 **menuSecondPhase** = **true**;  
 clearWindow();  
 secondMenuCreation();  
 **break**;  
  
 *// Option 2 as follows* **case "2"**:  
 *// Load character from file?* **menuFirstPhase** = **false**;  
 **menuSecondPhase** = **true**;  
 clearWindow();  
 secondMenuLoading();  
 **break**;  
 **case "LOAD"**:  
 **menuFirstPhase** = **false**;  
 **menuSecondPhase** = **true**;  
 clearWindow();  
 secondMenuLoading();  
 **break**;  
  
 *// Last exit option for total system shutdown* **case "0"**:  
 **myCharacterData**.saveFunc(**"characterData"**);  
 clearWindow();  
 System.***out***.println(**"\n\nSaving and exiting."**);  
 Thread.*sleep*(1000);  
 clearWindow();  
 System.***out***.println(**"\n\nSaving and exiting.."**);  
 Thread.*sleep*(1000);  
 clearWindow();  
 System.***out***.println(**"\n\nSaving and exiting..."**);  
 Thread.*sleep*(1000);  
 System.*exit*(0);  
 **break**;  
 **case "EXIT"**:  
 **myCharacterData**.saveFunc(**"characterData"**);  
 clearWindow();  
 System.***out***.println(**"\n\nSaving and exiting."**);  
 Thread.*sleep*(1000);  
 clearWindow();  
 System.***out***.println(**"\n\nSaving and exiting.."**);  
 Thread.*sleep*(1000);  
 clearWindow();  
 System.***out***.println(**"\n\nSaving and exiting..."**);  
 Thread.*sleep*(1000);  
 System.*exit*(0);  
 **break**;  
  
 **default**:  
 clearWindow();  
 System.***out***.println(**"\n\nTry again!"**);  
  
 }  
 }  
  
  
 */\* Second phase menu dynamic according to follow up choice  
 \* were checking of object concludes to avoid duplicate  
 \* creation, if loading initiated try object methods for  
 \* file reading and potentially writing... \*/* }  
  
 **public void** secondMenuCreation() **throws** IOException, InterruptedException {  
  
 clearWindow();  
 String nameToCheck = **"Empty"**;  
 String first = **"1"**;  
 String second = **"2"**;  
 String third = **"3"**;  
 String classPreviewString = **""**;  
 **validName** = **false**;  
  
 **while** (**menuSecondPhase**) {  
  
 Scanner scanner = **new** Scanner(System.***in***);  
  
 System.***out***.println(**"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"** +  
 **"\n|\t\t\t\t\t\t|"** +  
 **"\n|\t\tCreate Character\t\t|"** +  
 **"\n|\t\t\t\t\t\t|"** +  
 **"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\tName: "** + nameToCheck +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t["** + first + **"]Knight"** +  
 **"\n\t\t["** + second + **"]Wizard"** +  
 **"\n\t\t["** + third + **"]Thief"** +  
 **"\n"** +  
 **"\n"** + classPreviewString +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n[0]RETURN\t\t\t\t[X]SUBMIT\t"** +  
 **"\n\t\t\t\t\t\t"**);  
  
  
 String secondPhaseInput = scanner.nextLine().toUpperCase();  
 **if** (!secondPhaseInput.equals(**""**) && !secondPhaseInput.equals(**"1"**) &&  
 !secondPhaseInput.contains(**"2"**) && !secondPhaseInput.equals(**"3"**)  
 && !secondPhaseInput.equals(**"X"**) && !secondPhaseInput.equals(**"0"**)) {  
 nameToCheck = secondPhaseInput;  
 }  
  
 **switch** (secondPhaseInput) {  
  
 *// Test creation on submit for uniqueness* **case "1"**:  
 first = **"X"**;  
 second = **"2"**;  
 third = **"3"**;  
 classPreviewString = **myKnight**.previewLogo();  
 clearWindow();  
 **break**;  
 **case "2"**:  
 first = **"1"**;  
 second = **"X"**;  
 third = **"3"**;  
 classPreviewString = **myWizard**.previewLogo();  
 clearWindow();  
 **break**;  
 **case "3"**:  
 first = **"1"**;  
 second = **"2"**;  
 third = **"X"**;  
 classPreviewString = **myThief**.previewLogo();  
 clearWindow();  
 **break**;  
  
 *// Submit cases to try the given name to check object and break loop  
  
 // Object validation* **case "X"**:  
  
 **if** (first.equals(**"X"**)) {  
 **myKnight**.setName(nameToCheck);  
 **boolean** myBool = **myCharacterData**.addChar(**myKnight**);  
 **if** (myBool) {  
 System.***out***.println(**"\n\nSAVED!"**);  
 Thread.*sleep*(2000);  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 runMainMenu();  
 } **else** {  
 clearWindow();  
 System.***out***.println(**"\n\nNAME TAKEN!"**);  
 }  
 **break**;  
 }  
 **if** (second.equals(**"X"**)) {  
 **myWizard**.setName(nameToCheck);  
 **boolean** myBool = **myCharacterData**.addChar(**myWizard**);  
 **if** (myBool) {  
 System.***out***.println(**"\n\nSAVED!!"**);  
 Thread.*sleep*(2000);  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 runMainMenu();  
 } **else** {  
 clearWindow();  
 System.***out***.println(**"\n\nNAME TAKEN!"**);  
 }  
 **break**;  
 }  
 **if** (third.equals(**"X"**)) {  
 **myThief**.setName(nameToCheck);  
 **boolean** myBool = **myCharacterData**.addChar(**myThief**);  
 **if** (myBool) {  
 System.***out***.println(**"\n\nSAVED!"**);  
 Thread.*sleep*(2000);  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 runMainMenu();  
 } **else** {  
 clearWindow();  
 System.***out***.println(**"\n\nNAME TAKEN!"**);  
 }  
 **break**;  
 } **else** {  
 clearWindow();  
 System.***out***.println(**"\n\nCHOOSE CLASS!"**);  
 }  
 **break**;  
 **case "SUBMIT"**:  
  
 **if** (first.equals(**"X"**)) {  
 **myKnight**.setName(nameToCheck);  
 **boolean** myBool = **myCharacterData**.addChar(**myKnight**);  
 **if** (myBool) {  
 System.***out***.println(**"\n\nSAVED!"**);  
 Thread.*sleep*(2000);  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 } **else** {  
 clearWindow();  
 System.***out***.println(**"\n\nNAME TAKEN!"**);  
 }  
  
 **break**;  
 }  
 **if** (second.equals(**"X"**)) {  
 **myWizard**.setName(nameToCheck);  
 **boolean** myBool = **myCharacterData**.addChar(**myWizard**);  
 **if** (myBool) {  
 System.***out***.println(**"\n\nSAVED!"**);  
 Thread.*sleep*(2000);  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 } **else** {  
 clearWindow();  
 System.***out***.println(**"\n\nNAME TAKEN!"**);  
 }  
 **break**;  
 }  
 **if** (third.equals(**"X"**)) {  
 **myThief**.setName(nameToCheck);  
 **boolean** myBool = **myCharacterData**.addChar(**myThief**);  
 **if** (myBool) {  
 System.***out***.println(**"\n\nSAVED!"**);  
 Thread.*sleep*(2000);  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 } **else** {  
 clearWindow();  
 System.***out***.println(**"\n\nNAME TAKEN!"**);  
 }  
 **break**;  
 } **else** {  
 clearWindow();  
 System.***out***.println(**"\nCHOOSE CLASS!"**);  
 }  
 **break**;  
 *// Object validation  
  
  
 // Last exit option for total system shutdown* **case "0"**:  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 clearWindow();  
 runMainMenu();  
 **break**;  
 **case "RETURN"**:  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 clearWindow();  
 runMainMenu();  
 **break**;  
  
 **default**:  
  
 clearWindow();  
 System.***out***.println(**"\n\nSubmit to confirm!"**);  
  
 }  
 }  
 }  
  
 **public void** secondMenuLoading() **throws** IOException, InterruptedException {  
  
 clearWindow();  
 String nameToCheck = **"Empty"**;  
 **while** (**menuSecondPhase**) {  
  
 Scanner scanner = **new** Scanner(System.***in***);  
  
 System.***out***.println(**"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"** +  
 **"\n|\t\t\t\t\t\t|"** +  
 **"\n|\t\tLoad Character\t\t\t|"** +  
 **"\n|\t\t\t\t\t\t|"** +  
 **"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\tName: "** + nameToCheck +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n[0]RETURN\t\t\t\t[X]SUBMIT\t"** +  
 **"\n\t\t\t\t\t\t"**);  
  
  
 String choice = scanner.nextLine().toUpperCase();  
 **if** (!choice.equals(**""**) && !choice.equals(**"X"**)) {  
 nameToCheck = choice;  
 }  
 **switch** (choice) {  
  
 *// Submit cases to try the given name to check object and break loop* **case "X"**:  
 **myCharacter** = **myCharacterData**.searchCharacter(nameToCheck);  
 **if** (**myCharacter** != **null**) {  
 clearWindow();  
 System.***out***.println(**myCharacter**.previewLogo());  
 System.***out***.println(**"\n\nPress any key to continue..."**);  
 scanner.nextLine();  
 **menuFirstPhase** = **false**;  
 **menuSecondPhase** = **false**;  
 **menuThirdPhase** = **true**;  
 clearWindow();  
 thirdMenuGameLoader();  
 } **else** {  
 clearWindow();  
 System.***out***.println(**"\n\nCharacter doesn't exist!"**);  
 }  
 **break**;  
 **case "SUBMIT"**:  
 **myCharacter** = **myCharacterData**.searchCharacter(nameToCheck);  
 **if** (**myCharacter** != **null**) {  
 clearWindow();  
 System.***out***.println(**myCharacter**.previewLogo());  
 System.***out***.println(**"\n\nPress any key to continue..."**);  
 scanner.nextLine();  
 **menuFirstPhase** = **false**;  
 **menuSecondPhase** = **false**;  
 **menuThirdPhase** = **true**;  
 clearWindow();  
 thirdMenuGameLoader();  
 } **else** {  
 clearWindow();  
 System.***out***.println(**"\n\nCharacter doesn't exist!"**);  
 }  
 **break**;  
  
 **case "0"**:  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 clearWindow();  
 runMainMenu();  
 **break**;  
 **case "RETURN"**:  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 clearWindow();  
 runMainMenu();  
 **break**;  
  
 **default**:  
  
 clearWindow();  
 System.***out***.println(**"\n\nSubmit to confirm!"**);  
 }  
 }  
 }  
  
 **public void** thirdMenuGameLoader() **throws** IOException, InterruptedException {  
  
 clearWindow();  
 String first = **"1"**;  
 String second = **"2"**;  
 String third = **"3"**;  
 String tL = **"A"**;  
 String tR = **"B"**;  
 String bL = **"C"**;  
 String bR = **"D"**;  
  
  
  
 **while** (**menuThirdPhase**) {  
  
 Scanner scanner = **new** Scanner(System.***in***);  
  
  
 System.***out***.println(**"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"** +  
 **"\n|\t\t\t\t\t\t|"** +  
 **"\n|\t\tStart Game!\t\t\t|"** +  
 **"\n|\t\t\t\t\t\t|"** +  
 **"\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\tDIFFICULTY"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t["** + first + **"]Easy"** +  
 **"\n\t\t["** + second + **"]Medium"** +  
 **"\n\t\t["** + third + **"]Hard"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\tSPAWN\t\t\t\t"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t["** + tL + **"]Top L"** +  
 **"\n\t\t["** + tR + **"]Top R"** +  
 **"\n\t\t["** + bL + **"]Bottom L"** +  
 **"\n\t\t["** + bR + **"]Bottom R"** +  
 **"\n\t\t\t\t\t\t\t"** +  
 **"\n\t\t\t\t\t\t"** +  
 **"\n\t\t\t\t\t\t\t"** +  
 **"\n\t\t\t\t\t\t\t"** +  
 **"\n[0]RETURN\t\t\t\t[X]SUBMIT\t"** +  
 **"\n\t\t\t\t\t\t\t"**);  
  
  
 String thirdPhaseInput = scanner.nextLine().toUpperCase();  
  
 **switch** (thirdPhaseInput) {  
  
 *// Test creation on submit for uniqueness* **case "1"**:  
 first = **"X"**;  
 second = **"2"**;  
 third = **"3"**;  
 clearWindow();  
 **break**;  
 **case "2"**:  
 first = **"1"**;  
 second = **"X"**;  
 third = **"3"**;  
 clearWindow();  
 **break**;  
 **case "3"**:  
 first = **"1"**;  
 second = **"2"**;  
 third = **"X"**;  
 clearWindow();  
 **break**;  
 **case "A"**:  
 tL = **"X"**;  
 tR = **"B"**;  
 bL = **"C"**;  
 bR = **"D"**;  
 clearWindow();  
 **break**;  
 **case "B"**:  
 tL = **"A"**;  
 tR = **"X"**;  
 bL = **"C"**;  
 bR = **"D"**;  
 clearWindow();  
 **break**;  
 **case "C"**:  
 tL = **"A"**;  
 tR = **"B"**;  
 bL = **"X"**;  
 bR = **"D"**;  
 clearWindow();  
 **break**;  
 **case "D"**:  
 tL = **"A"**;  
 tR = **"B"**;  
 bL = **"C"**;  
 bR = **"X"**;  
 clearWindow();  
 **break**;  
  
  
 *// Submit cases to try the given name to check object and break loop  
  
 // Object validation* **case "X"**:  
  
 **if** (first.equals(**"X"**)) {  
 **submitDifficulty** = 1;  
 }  
 **if** (second.equals(**"X"**)) {  
 **submitDifficulty** = 2;  
 }  
 **if** (third.equals(**"X"**)) {  
 **submitDifficulty** = 3;  
 }  
  
 **if** (tL.equals(**"X"**)) {  
 **startingCorner** = 1;  
 }  
 **if** (tR.equals(**"X"**)) {  
 **startingCorner** = 2;  
 }  
 **if** (bL.equals(**"X"**)) {  
 **startingCorner** = 3;  
 }  
 **if** (bR.equals(**"X"**)) {  
 **startingCorner** = 4;  
 }  
  
  
 **if** (**submitDifficulty** > 0 && **startingCorner** > 0) {  
  
 *// Interact with object* **menuFirstPhase** = **false**;  
 **menuSecondPhase** = **false**;  
 **menuThirdPhase** = **false**;  
 clearWindow();  
 **break**;  
 }  
 **break**;  
 **case "SUBMIT"**:  
  
 **if** (first.equals(**"X"**)) {  
 **submitDifficulty** = 1;  
 }  
 **if** (second.equals(**"X"**)) {  
 **submitDifficulty** = 2;  
 }  
 **if** (third.equals(**"X"**)) {  
 **submitDifficulty** = 3;  
 }  
  
 **if** (tL.equals(**"X"**)) {  
 **startingCorner** = 1;  
 }  
 **if** (tR.equals(**"X"**)) {  
 **startingCorner** = 2;  
 }  
 **if** (bL.equals(**"X"**)) {  
 **startingCorner** = 3;  
 }  
 **if** (bR.equals(**"X"**)) {  
 **startingCorner** = 4;  
 }  
  
  
 **if** (**submitDifficulty** > 0 && **startingCorner** > 0) {  
  
 *// Interact with object* **menuFirstPhase** = **false**;  
 **menuSecondPhase** = **false**;  
 **menuThirdPhase** = **false**;  
 clearWindow();  
 **break**;  
  
  
 *// Implement next phase, game engine?* }  
 **break**;  
 *// Object validation* **case "0"**:  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 **menuThirdPhase** = **false**;  
 clearWindow();  
 runMainMenu();  
 **break**;  
 **case "RETURN"**:  
 **menuFirstPhase** = **true**;  
 **menuSecondPhase** = **false**;  
 **menuThirdPhase** = **false**;  
 clearWindow();  
 runMainMenu();  
 **break**;  
  
 **default**:  
  
 clearWindow();  
 System.***out***.println(**"Submit to confirm!"**);  
  
 }  
 }  
 }  
  
 *// Erases all text in terminal based on system and environment* **public void** clearWindow() **throws** IOException, InterruptedException {  
 **new** ProcessBuilder(**"cmd"**, **"/c"**, **"cls"**).inheritIO().start().waitFor();  
 }  
  
 **public boolean** isMenuFirstPhase() {  
 **return menuFirstPhase**;  
 }  
  
 **public void** setMenuFirstPhase(**boolean** menuFirstPhase) {  
 **this**.**menuFirstPhase** = menuFirstPhase;  
 }  
  
 **public boolean** isMenuSecondPhase() {  
 **return menuSecondPhase**;  
 }  
  
 **public void** setMenuSecondPhase(**boolean** menuSecondPhase) {  
 **this**.**menuSecondPhase** = menuSecondPhase;  
 }  
  
 **public boolean** isMenuThirdPhase() {  
 **return menuThirdPhase**;  
 }  
  
 **public void** setMenuThirdPhase(**boolean** menuThirdPhase) {  
 **this**.**menuThirdPhase** = menuThirdPhase;  
 }  
  
 **public** Character getMyCharacter() {  
 **return myCharacter**;  
 }  
  
 **public void** setMyCharacter(Character myCharacter) {  
 **this**.**myCharacter** = myCharacter;  
 }  
  
 **public** Knight getMyKnight() {  
 **return myKnight**;  
 }  
  
 **public void** setMyKnight(Knight myKnight) {  
 **this**.**myKnight** = myKnight;  
 }  
  
 **public** Wizard getMyWizard() {  
 **return myWizard**;  
 }  
  
 **public void** setMyWizard(Wizard myWizard) {  
 **this**.**myWizard** = myWizard;  
 }  
  
 **public** Thief getMyThief() {  
 **return myThief**;  
 }  
  
 **public void** setMyThief(Thief myThief) {  
 **this**.**myThief** = myThief;  
 }  
  
 **public** CollectionOfCharacters getMyCharacterData() {  
 **return myCharacterData**;  
 }  
  
 **public void** setMyCharacterData(CollectionOfCharacters myCharacterData) {  
 **this**.**myCharacterData** = myCharacterData;  
 }  
  
 **public** Adventure getMyAdventure() {  
 **return myAdventure**;  
 }  
  
 **public void** setMyAdventure(Adventure myAdventure) {  
 **this**.**myAdventure** = myAdventure;  
 }  
  
 **public int** getSubmitDifficulty() {  
 **return submitDifficulty**;  
 }  
  
 **public void** setSubmitDifficulty(**int** submitDifficulty) {  
 **this**.**submitDifficulty** = submitDifficulty;  
 }  
  
 **public int** getStartingCorner() {  
 **return startingCorner**;  
 }  
  
 **public void** setStartingCorner(**int** startingCorner) {  
 **this**.**startingCorner** = startingCorner;  
 }  
}

------

##### Text.java

------

**package** view;  
**import** java.io.IOException;  
**import** java.util.Scanner;  
  
**public class** Text {  
  
  
 **public void** intro() **throws** InterruptedException, IOException {  
  
 **boolean** state =**true**;  
 **int** time=0;  
  
 **while**(time!=2) {  
  
 **new** ProcessBuilder(**"cmd"**, **"/c"**, **"cls"**).inheritIO().start().waitFor();  
 System.***out***.println(Text.*introOverHeadCenter*());  
 Thread.*sleep*(200);  
  
 **new** ProcessBuilder(**"cmd"**, **"/c"**, **"cls"**).inheritIO().start().waitFor();  
 System.***out***.println(Text.*introOverHeadDown*());  
 Thread.*sleep*(200);  
  
 **new** ProcessBuilder(**"cmd"**, **"/c"**, **"cls"**).inheritIO().start().waitFor();  
 System.***out***.println(Text.*introOverHeadCenter*());  
 Thread.*sleep*(200);  
  
 **new** ProcessBuilder(**"cmd"**, **"/c"**, **"cls"**).inheritIO().start().waitFor();  
 System.***out***.println(Text.*introOverHeadUp*());  
 Thread.*sleep*(200);  
 time++;  
  
 }  
 *clearScreen*();  
  
 }  
  
  
 **public static void** pressToContinue(){  
 System.***out***.println(**"Press \"ENTER\" to continue..."**);  
 Scanner scanner = **new** Scanner(System.***in***);  
 scanner.nextLine();  
 }  
  
  
 **public static void** clearScreen() **throws** IOException, InterruptedException {  
 **new** ProcessBuilder(**"cmd"**, **"/c"**, **"cls"**).inheritIO().start().waitFor();  
 }  
  
  
 **public static** String introOverHeadDown() {  
 String overhead;  
  
  
 overhead =  
 **" \_\_\_\_\_\_\_ \_\_ \_\_ \_\_ \_\_ \_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_ \_\_ \_\_ \_\_\_\_\_\_\_ \_\_ \_\_ \_\_ \_\_ \n"** +  
 **"| \\ | \\ | \\| \\ | \\ / \\ | \\ / \\ | \\ | \\ | \\ | \\ | \\| \\ | \\\n"** +  
 **"| $$$$$$$\\| $$ | $$| $$\\ | $$| $$$$$$\\| $$$$$$$$| $$$$$$\\| $$\\ | $$ | $$$$$$$\\| $$ | $$| $$\\ | $$\n"** +  
 **"| $$ | $$| $$ | $$| $$$\\| $$| $$ \_\_\\$$| $$\_\_ | $$ | $$| $$$\\| $$ | $$\_\_| $$| $$ | $$| $$$\\| $$\n"** +  
 **"| $$ | $$| $$ | $$| $$$$\\ $$| $$| \\| $$ \\ | $$ | $$| $$$$\\ $$ | $$ $$| $$ | $$| $$$$\\ $$\n"** +  
 **"| $$ | $$| $$ | $$| $$\\$$ $$| $$ \\$$$$| $$$$$ | $$ | $$| $$\\$$ $$ | $$$$$$$\\| $$ | $$| $$\\$$ $$\n"** +  
 **"| $$\_\_/ $$| $$\_\_/ $$| $$ \\$$$$| $$\_\_| $$| $$\_\_\_\_\_ | $$\_\_/ $$| $$ \\$$$$ | $$ | $$| $$\_\_/ $$| $$ \\$$$$\n"** +  
 **"| $$ $$ \\$$ $$| $$ \\$$$ \\$$ $$| $$ \\ \\$$ $$| $$ \\$$$ | $$ | $$ \\$$ $$| $$ \\$$$\n"** +  
 **" \\$$$$$$$ \\$$$$$$ \\$$ \\$$ \\$$$$$$ \\$$$$$$$$ \\$$$$$$ \\$$ \\$$ \\$$ \\$$ \\$$$$$$ \\$$ \\$$\n"** +  
 **" \n"** +  
 **"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"**+  
 **"Av: Kim Burhan Peter Axel Kassim "**;  
  
  
 **return** overhead;  
 }  
  
 **public static** String introOverHeadUp() {  
 String overhead;  
  
  
 overhead = **"\n"** +  
 **"$$$$$$$\\ $$\\ $$\\ $$\\ $$\\ $$$$$$\\ $$$$$$$$\\ $$$$$$\\ $$\\ $$\\ $$$$$$$\\ $$\\ $$\\ $$\\ $$\\ \n"** +  
 **"$$ \_\_$$\\ $$ | $$ |$$$\\ $$ |$$ \_\_$$\\ $$ \_\_\_\_\_|$$ \_\_$$\\ $$$\\ $$ | $$ \_\_$$\\ $$ | $$ |$$$\\ $$ |\n"** +  
 **"$$ | $$ |$$ | $$ |$$$$\\ $$ |$$ / \\\_\_|$$ | $$ / $$ |$$$$\\ $$ | $$ | $$ |$$ | $$ |$$$$\\ $$ |\n"** +  
 **"$$ | $$ |$$ | $$ |$$ $$\\$$ |$$ |$$$$\\ $$$$$\\ $$ | $$ |$$ $$\\$$ | $$$$$$$ |$$ | $$ |$$ $$\\$$ |\n"** +  
 **"$$ | $$ |$$ | $$ |$$ \\$$$$ |$$ |\\\_$$ |$$ \_\_| $$ | $$ |$$ \\$$$$ | $$ \_\_$$< $$ | $$ |$$ \\$$$$ |\n"** +  
 **"$$ | $$ |$$ | $$ |$$ |\\$$$ |$$ | $$ |$$ | $$ | $$ |$$ |\\$$$ | $$ | $$ |$$ | $$ |$$ |\\$$$ |\n"** +  
 **"$$$$$$$ |\\$$$$$$ |$$ | \\$$ |\\$$$$$$ |$$$$$$$$\\ $$$$$$ |$$ | \\$$ | $$ | $$ |\\$$$$$$ |$$ | \\$$ |\n"** +  
 **"\\\_\_\_\_\_\_\_/ \\\_\_\_\_\_\_/ \\\_\_| \\\_\_| \\\_\_\_\_\_\_/ \\\_\_\_\_\_\_\_\_| \\\_\_\_\_\_\_/ \\\_\_| \\\_\_| \\\_\_| \\\_\_| \\\_\_\_\_\_\_/ \\\_\_| \\\_\_|\n"** +  
 **" \n"** +  
 **"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"**+  
 **"Av: Kim Burhan Peter Axel Kassim "**;  
  
  
 **return** overhead;  
 }  
  
 **public static** String introOverHeadCenter() {  
 String overhead;  
  
 overhead = **"\n"** +  
 **" /$$$$$$$ /$$ /$$ /$$ /$$ /$$$$$$ /$$$$$$$$ /$$$$$$ /$$ /$$ /$$$$$$$ /$$ /$$ /$$ /$$\n"** +  
 **"| $$\_\_ $$| $$ | $$| $$$ | $$ /$$\_\_ $$| $$\_\_\_\_\_/ /$$\_\_ $$| $$$ | $$ | $$\_\_ $$| $$ | $$| $$$ | $$\n"** +  
 **"| $$ \\ $$| $$ | $$| $$$$| $$| $$ \\\_\_/| $$ | $$ \\ $$| $$$$| $$ | $$ \\ $$| $$ | $$| $$$$| $$\n"** +  
 **"| $$ | $$| $$ | $$| $$ $$ $$| $$ /$$$$| $$$$$ | $$ | $$| $$ $$ $$ | $$$$$$$/| $$ | $$| $$ $$ $$\n"** +  
 **"| $$ | $$| $$ | $$| $$ $$$$| $$|\_ $$| $$\_\_/ | $$ | $$| $$ $$$$ | $$\_\_ $$| $$ | $$| $$ $$$$\n"** +  
 **"| $$ | $$| $$ | $$| $$\\ $$$| $$ \\ $$| $$ | $$ | $$| $$\\ $$$ | $$ \\ $$| $$ | $$| $$\\ $$$\n"** +  
 **"| $$$$$$$/| $$$$$$/| $$ \\ $$| $$$$$$/| $$$$$$$$| $$$$$$/| $$ \\ $$ | $$ | $$| $$$$$$/| $$ \\ $$\n"** +  
 **"|\_\_\_\_\_\_\_/ \\\_\_\_\_\_\_/ |\_\_/ \\\_\_/ \\\_\_\_\_\_\_/ |\_\_\_\_\_\_\_\_/ \\\_\_\_\_\_\_/ |\_\_/ \\\_\_/ |\_\_/ |\_\_/ \\\_\_\_\_\_\_/ |\_\_/ \\\_\_/\n"** +  
 **" \n"** +  
 **"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"**+  
 **"Av: Kim Burhan Peter Axel Kassim "**;  
  
  
 **return** overhead;  
 }  
  
  
}

------

##### JsonUtil.java

-----

**import** model.Character.Thief;  
  
**import** java.io.\*;  
  
**public class** JsonUtil {  
 **public static void** main(String[] args) {  
 *serlize*();  
 }  
 **private static void** serlize() {  
 Thief char1 = **new** Thief();  
 Thief char2 = **new** Thief();  
  
  
 */\*try(FileWriter file = new FileWriter("file1.txt")){  
 file.write(json);  
 file.write(json2);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }\*/* String fileContent=**""**;  
 **try** {  
 File f = **new** File(**"file1.txt"**);  
 FileInputStream inp = **new** FileInputStream(f);  
 **byte**[] bf = **new byte**[(**int**)f.length()];  
 inp.read(bf);  
 fileContent = **new** String(bf, **"UTF-8"**);  
 } **catch** (FileNotFoundException e) {  
 e.printStackTrace();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 System.***out***.println(fileContent);  
  
  
  
 }  
}

---------

##### Main.java

------

**import** model.Adventure;  
**import** model.Character.\*;  
**import** model.Character.Character;  
**import** model.Character.Knight;  
**import** model.Character.Thief;  
**import** model.Monster.Monster;  
**import** model.Monster.Skeleton;  
**import** view.Combat;  
*//import view.Menu;***import** view.Menu;  
**import** view.Text;  
  
**import** java.io.IOException;  
**import** java.util.ArrayList;  
  
**public class** Main {  
  
 **public static void** main(String[] args) **throws** IOException, InterruptedException {  
 *// Testing execution flow from this point* Text credits = **new** Text();  
 credits.intro();  
  
 Menu mainMenu = **new** Menu();  
 mainMenu.runMainMenu();  
  
 Adventure gameEngine = **new** Adventure(mainMenu);  
 gameEngine.startNewRoom();  
  
  
  
 }  
}

-----

