import java.security.SecureRandom;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

public class BoardImpl implements Board{

Tile[] board = new Tile[101]; //A array to represent the 'board'

List<Player> players = new ArrayList<>();

SecureRandom secureRandom = new SecureRandom();

Set<Integer> occupiedPositions = new HashSet<>();

public void setPlayers(List<Player> players) {

this.players = players;

}

public Tile[] getBoard() {

return board;

}

public List<Player> getPlayers() {

return players;

}

public void addPlayers(Player player) {

this.players.add(player);

}

@Override

public void initialize() {

//this function generates the board, placing random snakes and ladders and

//filling the rest with normal tiles

System.out.println("Initializing board");

int snakes = secureRandom.nextInt(15) + 1; //generates snakes &

int ladders = secureRandom.nextInt(15) + 1; //ladders at random

//positions

setSnakes(snakes);

setLadders(ladders);

for(int i = 0; i < 101; i++){

if(board[i] != null){

continue;

}

Tile t = new TileImpl(TileType.NORMAL);

board[i] = t;

}

}

@Override

public void movePlayer(Player player, int places) {

int currentPosition = player.getPosition();

String checkPos = checkEndingPosition(currentPosition, places);

if(checkPos.equals("winner")){

winner(player);

}

if(checkPos.equals("outOfBoard")){

System.out.println("That move cannot be made from your position! " +

"Roll a different number next time");

}

if(checkPos.equals("valid")){

Tile t = getTile(currentPosition + places);

if(t.getType().equals(TileType.NORMAL)){

player.setPosition(currentPosition + places);

System.out.println("Player " + player.getName() +

" moves to position "+ player.getPosition());

}else {

player.setPosition(t.getDestination());

System.out.println(" Player " + player.getName() +

" encountered a "+ t.getType()+ "!!, moves to position "+ player.getPosition());

}

}

}

@Override

public Tile getTile(int n) {

return board[n];

}

public String checkEndingPosition(int currentPosition, int places){

if((currentPosition + places) > 100){

return "outOfBoard";

}else if((currentPosition+ places) == 100){

return "winner";

}

else{

return "valid";

}

}

public void winner(Player player){

System.out.print("Player " + player.getName() + " has won the game !");

System.exit(0);

}

public void setSnakes(int n){

//this function generates snakes at random positions

for(int i = 0; i < n; i ++){

boolean flag = true;

int start = 0;

int dest = 0;

while(flag) {

start = secureRandom.nextInt(98);

if(!occupiedPositions.contains(start)){

occupiedPositions.add(start);

break;

}

}

while(flag) {

//Setting the destination for a snake tile to lower than its position

dest = (int)(start \* secureRandom.nextDouble());

if(!occupiedPositions.contains(dest)){

occupiedPositions.add(dest);

break;

}

}

Tile tile = new TileImpl(TileType.SNAKE, dest);

board[start] = tile;

System.out.println("Created snake " + "[" + start+ "," + dest + "]");

}

}

public void setLadders(int n) {

//this function generates ladders randomly

for (int i = 0; i < n; i++) {

boolean flag = true;

int start = 0;

int dest = 0;

while (flag) {

start = secureRandom.nextInt(80);

if (!occupiedPositions.contains(start)) {

occupiedPositions.add(start);

break;

}

}

while (flag) {

//this step places the destination for the ladder tile to greater than its position

dest = (int) (start + secureRandom.nextDouble() \* 10);

if (!occupiedPositions.contains(dest)) {

occupiedPositions.add(dest);

break;

}

}

Tile tile = new TileImpl(TileType.LADDER, dest);

board[start] = tile;

System.out.println("Created ladder " + "[" + start+ "," + dest + "]");

}

}

}

public class PlayerImpl implements Player{

private int position = 0;

private String name = null;

public PlayerImpl(String name){

this.name = name;

}

@Override

public int getPosition() {

return position;

}

@Override

public String getName() {

return name;

}

@Override

public void setPosition(int currentPosition) {

this.position = currentPosition;

}

@Override

public void setName(String playerName) {

this.name = playerName;

}

}

public class TileImpl implements Tile {

private TileType type;

private int position;

private int destination;

public TileImpl(TileType type) {

this.type = type;

}

public TileImpl(TileType type, int dest) {

this.type = type;

this.destination = dest;

}

public int getPosition() {

return position;

}

public int getDestination() {

return destination;

}

public void setType(TileType type) {

this.type = type;

}

public void setPosition(int position) {

this.position = position;

}

public void setDestination(int destination) {

this.destination = destination;

}

@Override

public TileType getType() {

return type;

}

}

import java.security.SecureRandom;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class SnakesAndLadders {

public static void main(String args[]){

Board board = new BoardImpl();

List<Player> players = new ArrayList<>();

System.out.println("How many players ?");

Scanner scan = new Scanner(System.in);

int noOfPlayers = scan.nextInt();

for(int i = 0; i < noOfPlayers; i++){

System.out.println("Enter name for player "+ i);

String name = scan.next();

Player player = new PlayerImpl(name);

players.add(player);

}

board.initialize();

int counter = 0;

System.out.println("Lets Play!");

String choice = "";

SecureRandom random = new SecureRandom();

//simulating a game through the do while loop

do{

if(counter >= noOfPlayers) counter = 0;

Player currPlayer = players.get(counter);

System.out.println(" Player " + currPlayer.getName() + " turn to play!");

System.out.println(" R = Roll the dice, Q = quit");

choice = scan.next();

if(choice.equalsIgnoreCase("R")){

int places = random.nextInt(6) + 1;

board.movePlayer(currPlayer, places);

counter++;

}

}while(!choice.equalsIgnoreCase("Q"));

}

}