**Програм хангамжийн хөгжүүлэлт**

(Дадлага ажил 1)

ХШУИС, Програм хангамжийн 3 курсын оюутан А.Дэмбэрэлдодов 20B1NUM0369

# Оршил

Энэхүү дадлага ажил 1 хүрээнд жава хэлтэй танилцах болон argoUML дээр классын диаграм код руу хөрвүүлэх.

# Зорилго

Лекцүүдийн хүрээнд үзэх шаардлагатай агуулгуудыг судлаж, өөрийн үгээр тайлбарлаж тэмдэглэл бичих. Номны дасгалыг уншиж, ойлгож ажиллах. Зорилгодоо хүрэхийн тулд дараах зорилтуудыг дэвшүүлсэн байгаа.

1. Лекцийн материал болон лекцийн үзэж ойлгох

Дасгал 2) (4 оноо)

Дасгал 1-д гүйцэтгэсэн “Hello world” програмыг өөрчлөн “Hello XXX” болгон хэвлэдэг болгоно. Энд XXX нь таны нэр байх болно. Програмын үр дүнгийн дэлгэцийн зургийг баримтад хавсаргана.

public class lab1 {

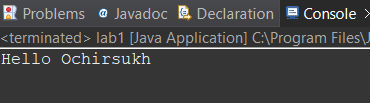
public static void main(String[] args) {

System.***out***.println("Hello Ochirsukh");

}

}

Үр дүн:



3) Жавагийн гарын авлага “A Closer Look at the "Hello World!" Application” –ийн “Source Code Comments” хэсгийг уншина уу. Өмнөх асуулт дээр үүсгэсэн Hello.java класст гүйцэтгэж буй үйл ажиллагаанд тохирох баримтын тайлбар үүсгэнэ (монгол хэлээр галиглаж бичнэ). Hello.java файлын бүх агуулгыг хуулбарлан баримтад хавсаргана.

// lab1 class todorhoilson bna

public class lab1 {

// main function shaardlagatai bvh function duudah bolomjtoi bolno

public static void main(String[] args) {

// delgets ruu text hewlej bna

System.***out***.println("Hello Ochirsukh");

}

(4 оноо) “Permuter” нэртэй классыг агуулсан permuter нэртэй төсөл (прожект) үүсгэнэ. Eclipse-ээр автоматаар үүсгэсэн кодыг дараах Python форматаар өгөгдсөн кодоор сольж бичнэ.

public class Permuter {

public static void main(String[] args) {

*permute*(Integer.*parseInt*(args[0]), args[1].toCharArray());

}

private static void permute(int n, char[] a) {

if (n == 0)

System.***out***.println(String.*valueOf*(a));

else

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

*permute*(n - 1, a);

*swap*(a, n % 2 == 0 ? i : 0, n);

}

}

private static void swap(char[] a, int i, int j) {

char saved = a[i];

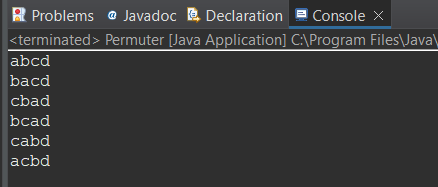
a[i] = a[j];

a[j] = saved;

}

}

Үр дүн

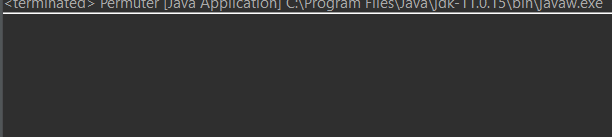


1. Таны сонгосон програмын аргумент юу вэ?

Int утга = -1;

Char aa[] = {‘a’, ‘d’, ‘b’};

1. Result



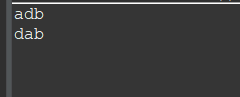
Яагаад дээрх үр дүн үзүүлсэн бэ гэвэл permute method дээрх – утга шалгадаггүй нөхцөл байхгүй. 0 тэнцүү биш бол else тохиолдолд очно. Дараа I =0 эхлүүлээд – утгатай жишиж байгаа тул ямар ч үр дүн үзүүлэхгүй. Энэ утга бол сөрөг жишээ үүсгэх зорилгоор ашигласан юм.

Result 2 :

Оруулсан утга:

Int утга = 1;

Char aa[] = {‘a’, ‘d’, ‘b’};



Permute method руу 1 болон char хүснэгт очно. 1 утга 0 тэй тэнцүү биш учир рекурсиваар дахин дуудаж 0 болж adb гэсэн утгийг хэвлэнэ. Дараа нь swap n = 1;

I = 0 дуудагдана swap руу а хүснэгт, 0, 1 гэсэн утга дамжуулж dab гэсэн утга болж өөрчлөгдөнө. I = 1 болно n = 1 дахин дуудагдаж dab гэсэн утгийг хэвлэнэ. I = 1 n = 1 adb болно. I =2 n = 1 их учир програм зогсож байна.

5) (4 оноо) Эхлээд дараах холбоосноос класс гэж юу вэ? Объект гэж юу вэ? гэдгийг эндээс уншиж танилцана. Мөн Variable болон Operator-ийн талаар уншаарай.

import java.util.Scanner;

public class TempConv {

public static void main(String arg[]) {

double fahr;

double cel;

Scanner in;

in = new Scanner(System.***in***);

System.***out***.println("Enter the temperature in F: ");

fahr = in.nextDouble();

cel = (fahr - 32) \* 5.0/9.0;

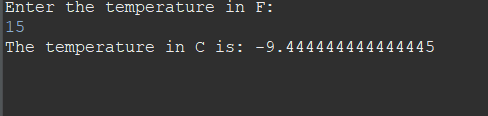
System.***out***.println("The temperature in C is: " + cel);

System.*exit*(0);

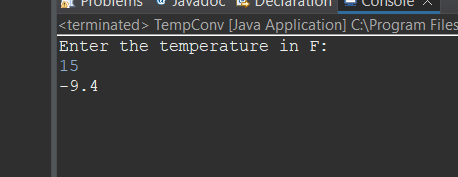
}

}

Result:



b. Result:



1. Өмнөх Section 4.3-ийг үргэлжлүүлэн уншина. TempConvGUI гэсэн Swing програмыг олж Eclipse-д үүсгэн ажиллуулна.

import javax.swing.\*;

/\*\*

\* **@author** user

\*

\*/

public class TempGUI {

public static void main(String[] args) {

String fahrString;

double fahr, cel;

int n = 1;

while(n != 0){

fahrString = JOptionPane.*showInputDialog*("Enter the temperature in F");

fahr = Double.*parseDouble*(fahrString);

if(fahr == 0) {

n = 0;

break;

}

cel = (fahr - 32) \* 5.0 / 9.0;

JOptionPane.*showMessageDialog*(null, "The temperature in C is, " + cel);

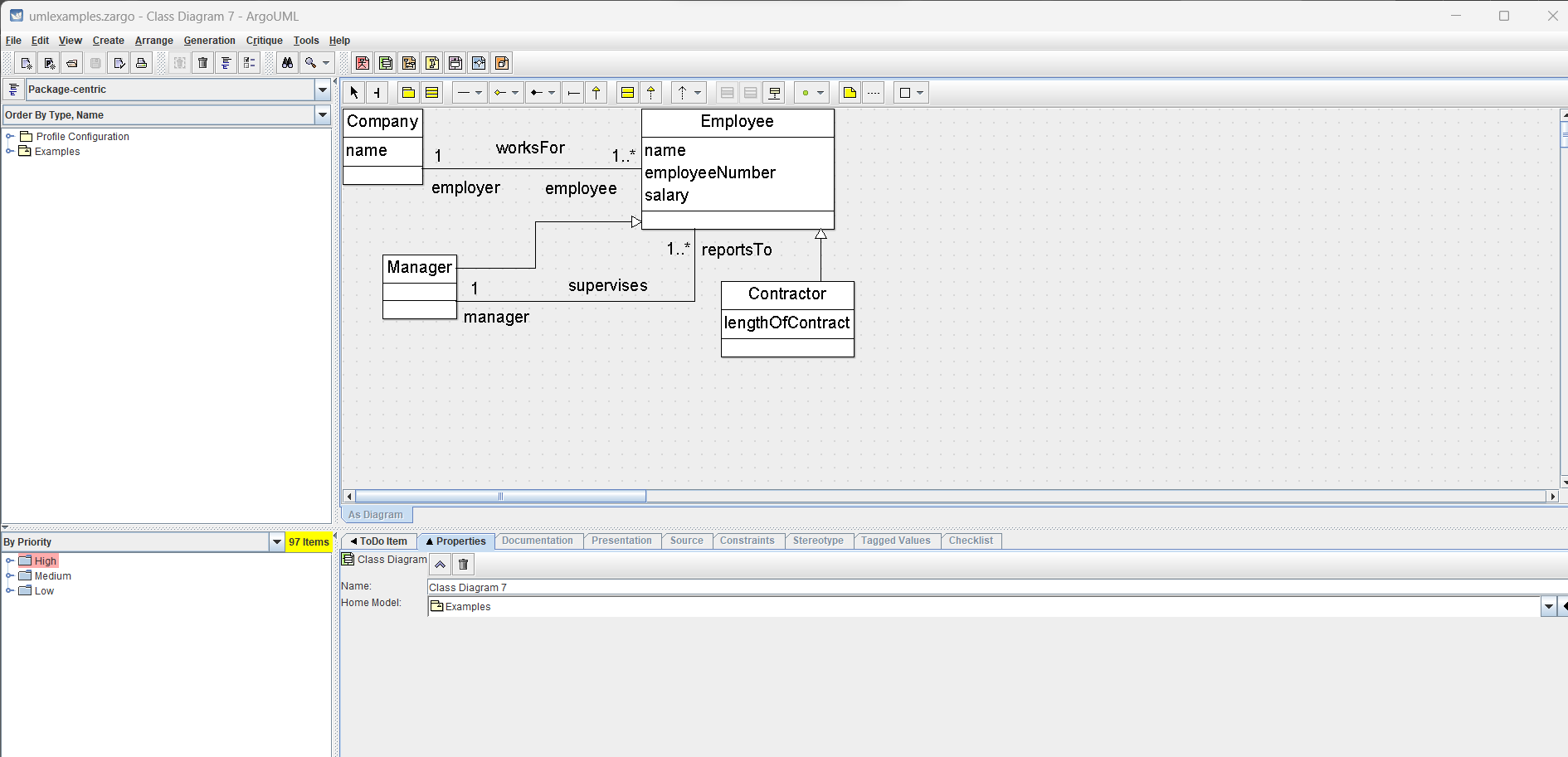
}

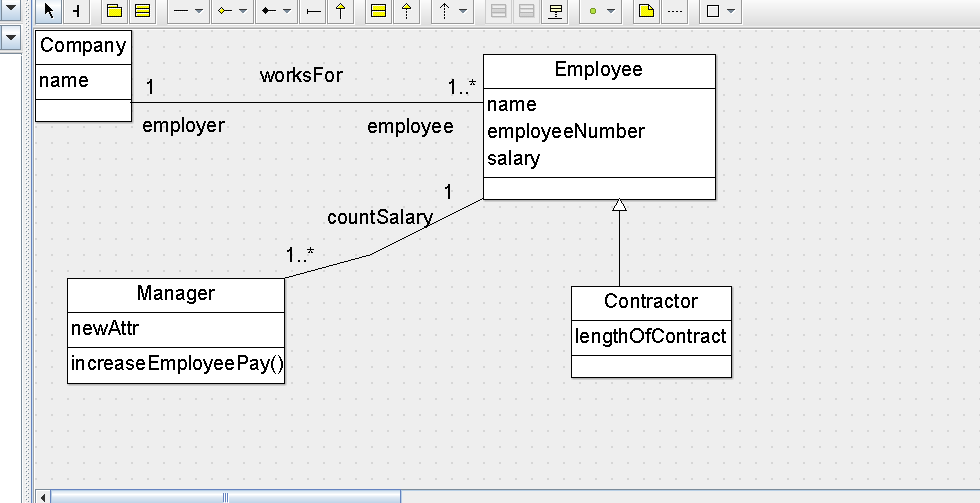
}

}

Result:







1. Game.java

package hucklebuckle;

class Game {

private Hider myHider;

private Seeker mySeeker;

private int gridSize;

Game(int gs) {

myHider = new Hider("Harry", this);

mySeeker = new Seeker("Sally", this);

gridSize = gs;

}

int getGridSize() {

return gridSize;

}

void play() {

System.***out***.println("Playing HuckleBuckle v" + HuckleBuckle.***VERSION***

+ " on a " + gridSize + " by " + gridSize + " grid...");

myHider.introduceYourself();

mySeeker.sayHelloTo(myHider); // let Sally know who will answer her

// questions

myHider.hide(); // tell Harry to hide an object

mySeeker.seek(); // tell Sally to start looking

}

}

Hider.Java

package hucklebuckle;

import java.lang.Math;

class Hider extends Player {

Seeker mySeeker;

Hider(String name, Game game) {

super(name, game);

}

void introduceYourself() {

System.***out***.println(" Harry says \"Hi, I'm " + getName()

+ ", let's play Huckle Buckle!\"");

}

void hide() { // move to a random place

setX((int) (Math.*random*() \* getGame().getGridSize()));

setY((int) (Math.*random*() \* getGame().getGridSize()));

}

/\*\*

\* **@param** seeker

\* the other player on the 2d grid

\* **@return** a Euclidean distance from the Hider to the seeker, in the range

\* [0.0, sqrt(2)\*gridSize]

\*/

private double distance(Player seeker) {

return Math.*hypot*(getX() - seeker.getX(), getY() - seeker.getY());

}

/\*\*

\* **@param** s

\* a Seeker `

\* **@return** 0, if the Seeker has found the hidden object. Returns a larger

\* int-valued "temperature" if the Seeker is farther away.

\*/

int revealTemperature(Seeker s) {

// compute a scaled distance. The range of r is 0.0 to 1.414

double r = distance(s) / (double) getGame().getGridSize();

System.***out***.print(" " + getName() + " says to " + s.getName() + ", \"");

if (r == 0.0) {

System.***out***.println("Huckle buckle beanstalk!\"");

return 0;

} else if (r <= 0.25) {

System.***out***.println("You're boiling!\"");

return 1;

} else if (r <= 0.35) {

System.***out***.println("You're hot.\"");

return 2;

} else if (r <= 0.5) {

System.***out***.println("You're warm.\"");

return 3;

} else if (r <= 0.7) {

System.***out***.println("You're cool.\"");

return 4;

} else if (r <= 1.0) {

System.***out***.println("You're cold.\"");

return 5;

} else {

System.***out***.println("freezing!\"");

return 6;

}

}

}

HuckleBuckle.java

package hucklebuckle;

/\*\*

\* A non-interactive console application which simulates a children's game.

\* <p>

\* Optional command-line parameter: gridsize, the size of the playing field, an

\* integer between 1 and 40 inclusive.

\* <p>

\* When versioning this code, programmers should adjust the value of

\* HuckleBuckle.VERSION so that an accurate version number appears on the first

\* line of program output.

\*

\* <p>

\* Second optional command-line parameter: ngames, the number of games to be

\* played on this run of the program, an integer between 0 and 9 inclusive.

\* Note: if only one command-line parameter is supplied, it is interpreted as

Програм хангамжийн хөгжүүлэлт ICSI402

6

\* gridsize rather than as ngames, as indicated by the nested bracketing 9

\* on the usage comment below.

\* <p>

\* Usage: hbbv1.1 [gridsize [ngames]]

\*

\* **@author** Clark Thomborson

\*

\*/

public class HuckleBuckle {

public static final String ***VERSION*** = "1.1";

public static void main(String[] args) {

int gridSize = 5; // gridSize == 5, if there are no command-line args

if (args.length > 0) {

try {

gridSize = Integer.*parseInt*(args[0]);

} catch (NumberFormatException e) {

System.***err***

.println("Error: first arg (gridSize) must be an integer. ");

System.*exit*(1);

}

if (gridSize < 1 || gridSize > 40) {

System.***err***

.println("Error: first arg (gridSize) must be in the range 1..40. ");

System.*exit*(1);

}

}

if (args.length > 1) {

System.***err***.println("Warning: too many args. ");

}

System.***out***.println("TOGLOOM EHLEJ BNA.");

// for(int i=0 ; i<Integer.parseInt(args[1]); i++) {

Game myGame = new Game(gridSize);

myGame.play();

//}

System.***out***.println("TOGLOOM DUUSLAA.");

}

}

Player.java

package hucklebuckle;

class Player {

private String myName;

private Game myGame;

private int x; // constraint: 0 <= x < HuckleBuckle.GridSize

private int y; // constraint: 0 <= y < HuckleBuckle.GridSize

Player(String name, Game game) {

myName = name;

myGame = game;

setX(0); // all players start at (0,0)

setY(0);

}

String getName() {

return myName;

}

Game getGame() {

return myGame;

}

int getX() {

return x;

}

void setX(int px) {

x = (int) px; // **TODO**: add bounds-checking logic

}

int getY() {

return y;

}

void setY(int py) {

y = (int) py; // **TODO**: add bounds-checking logic

}

}

Seeker.java

package hucklebuckle;

class Seeker extends Player {

private Hider myHider;

private int distanceMoved; // updated after every change of position

Seeker(String name, Game game) {

super(name, game);

distanceMoved = 0;

}

void sayHelloTo(Hider h) {

myHider = h;

System.***out***.println(getName() + " says, \"Hi, " + h.getName() + ", I'm "

+ getName() + ". Glad to meet you!\"");

}

private void reportLocation() {

System.***out***.println(getName() + " says, \"I'm at " + getX() + ", "

+ getY() + ".\"");

}

void seek() {

reportLocation(); // I report my initial location

boolean moveRight = true; // I move left-to-right on even-numbered rows

for (int y = 0; y < getGame().getGridSize(); y++) {

if (getY() != y) {

setY(y); // I move to row y, if I'm not already there

distanceMoved++;

reportLocation(); // I announce my location after every move

}

for (int cx = 0; cx < getGame().getGridSize(); cx++) {

// cx: counts the number of times I have moved on row y

int x = (moveRight ? cx : getGame().getGridSize() - cx - 1);

if (getX() != x) {

setX(x);

distanceMoved++;

reportLocation(); // I announce my location after every move

}

if (myHider.revealTemperature(this) == 0) { // Did I find it?

System.***out***.println(getName()

+ " says, \"That was fun! I walked "

+ distanceMoved + " steps before I found it.\"");

return; // I stop when I find the hidden object

}

}

// I move in the opposite direction on each row of the grid

moveRight = !moveRight;

// My movement pattern is sometimes called the snake-like row-major

// ordering. See http://en.wikipedia.org/wiki/Boustrophedon.

}

System.***out***.println(getName() + " says, \"I'm giving up. I took "

+ distanceMoved + "steps before quitting.\"");

}

}

1. A

Hbbv1.1-ийн эх кодыг тайлбарлан оруулна

5 класс агуулагдаж байгаа.

**HuckleBuckle.java**: тоглоомын талбай gridPane утга болон тоглоом хэдэн удаа давтах бэ? гэсэн утгийг авна. Тоглоомын талбай байгуулаад Game классын объектыг үүсгээд play() method-ийг дуудаж тоглоомоо эхлэнэ.

**Game.java:** тоглогчдын нэрс болон тоглоомын талбайн хэмжээг авна. Player1 болон Player2 хоорондоо мэндлэнэ. Дараа нь Player1 нь тоглоомын талбайд өөрийн random-оор байршилаа нууна. Player2 бол түүний байршилийг олох зорилгоор тоглож байгаа.

**Hide.java:** Player1 нуугдах талбайгаатаамгаар байршуулна.

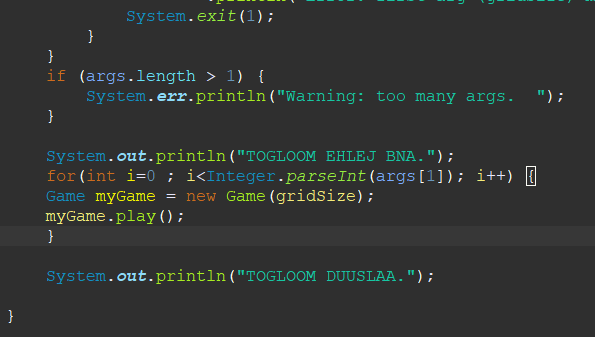
**Seeker буюу Player2 ийн байршилаас хэр хол байгааг илэрхийлнэ.** Зайнаасаа хамаарч temperature байдлаар өөртөө ойртож байгааг илэрхийлнэ. Хэрвээ олдвол game дуусна.

**Seeker.**java: Player2 одоо байгаа байршилийг авж х, y тэнхлэгийн дагуу явж

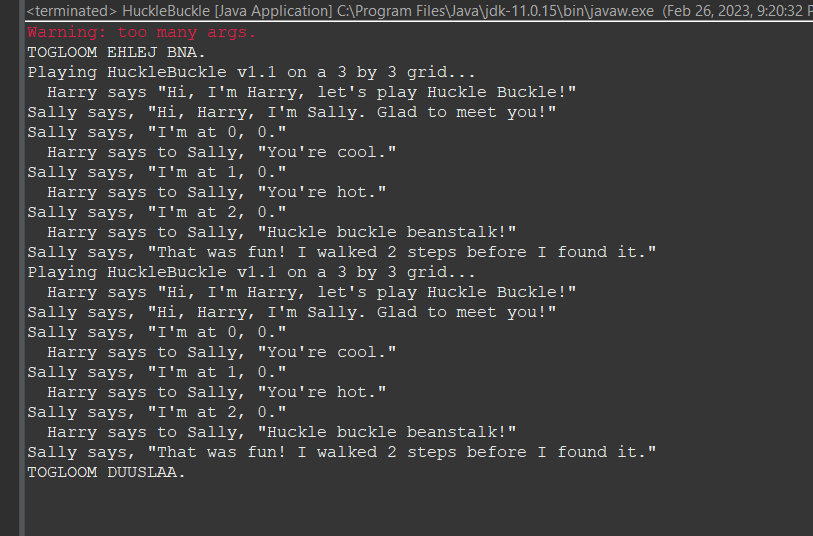
**Player1 байршилийг хайж олно.**

Player.java бол тоглоомын талбар x, y тэнхлэг, нэр зэрэг бүхий л утгийг хадгална.

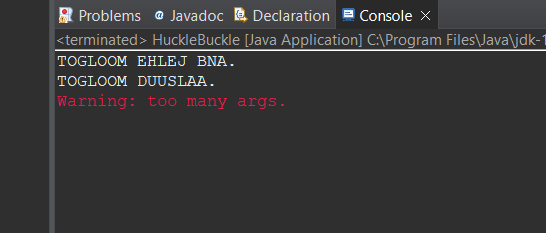
Тоглоомыг ngames-ийн тоогоор давтан тоглуулдаг болгосон давталтыг харуулна



Тус програмд 3 2 гэсэн аргумент дамжуулан өгч ажилласан үр дүнг оруулна.

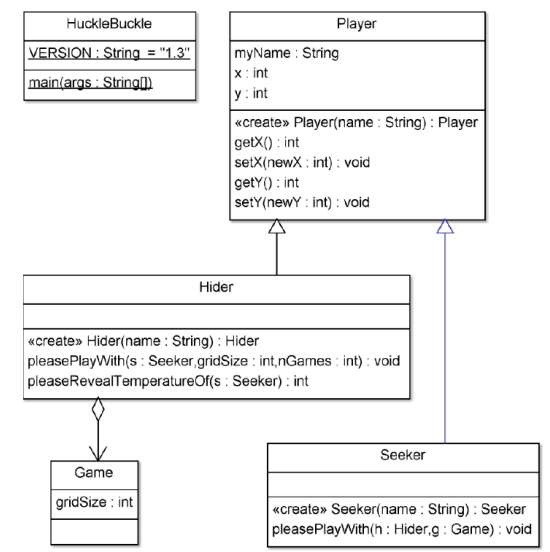


Тус програмд 4 0 гэсэн аргумент дамжуулан өгч ажилласан үр дүнг оруулна.



1. package hucklebuckle;
2. class Seeker extends Player {
3. private Hider myHider;
4. private int distanceMoved; // updated after every change of position
5. Seeker(String name, Game game) {
6. super(name, game);
7. distanceMoved = 0;
8. }
9. void sayHelloTo(Hider h) {
10. myHider = h;
11. System.***out***.println(getName() + " says, \"Hi, " + h.getName() + ", I'm "
12. + getName() + ". Glad to meet you!\"");
13. }
14. private void reportLocation() {
15. System.***out***.println(getName() + " says, \"I'm at " + getX() + ", "
16. + getY() + ".\"");
17. }
18. void moveTo(int x, int y) {
19. if (getY() != y) {
20. setY(y); // I move to row y, if I'm not already there
21. distanceMoved++;
22. reportLocation(); // I announce my location after every move
23. }
24. else if (getX() != x) {
25. setX(x); // I move to row y, if I'm not already there
26. distanceMoved++;
27. reportLocation(); // I announce my location after every move
28. }
29. else{
30. double r = this.distance() / (double) getGame().getGridSize();
31. if(myHider.revealTemperature(this, r) == 0)
32. System.***out***.println(getName()
33. + " says, \"That was fun! I walked "
34. + distanceMoved + " steps before I found it.\"");
35. return; // I stop when I find the hidden object
36. }
37. }
38. void seek() {
39. reportLocation(); // I report my initial location
40. boolean moveRight = true; // I move left-to-right on even-numbered rows
41. for (int y = 0; y < getGame().getGridSize(); y++) {
42. for (int cx = 0; cx < getGame().getGridSize(); cx++) {
43. // cx: counts the number of times I have moved on row y
44. int x = (moveRight ? cx : getGame().getGridSize() - cx - 1);
45. moveTo(x, y);
46. }
47. // I move in the opposite direction on each row of the grid
48. moveRight = !moveRight;
49. // My movement pattern is sometimes called the snake-like row-major
50. // ordering. See http://en.wikipedia.org/wiki/Boustrophedon.
51. }
52. System.***out***.println(getName() + " says, \"I'm giving up. I took "
53. + distanceMoved + "steps before quitting.\"");
54. }
55. }

12.



Game.java

package hucklebuckle;

class Game {

protected int gridSize;

Game(int gs) {

gridSize = gs;

}

int getGridSize() {

return gridSize;

}

}

Hider.java

package hucklebuckle;

import java.lang.Math;

class Hider extends Player {

Seeker mySeeker;

Game game = new Game(4);

int a = game.getGridSize();

Hider(String name, Game game) {

super(name, game);

}

void introduceYourself() {

System.***out***.println(" Harry says \"Hi, I'm " + getName()

+ ", let's play Huckle Buckle!\"");

}

void pleasePlayWith(Seeker seeker, int a, int b ) {

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

seeker.sayHelloTo(this);

seeker.seek();

}

}

void hide() { // move to a random place

setX((int) (Math.*random*() \* getGame().getGridSize()));

setY((int) (Math.*random*() \* getGame().getGridSize()));

}

/\*\*

\* **@param** seeker

\* the other player on the 2d grid

\* **@return** a Euclidean distance from the Hider to the seeker, in the range

\* [0.0, sqrt(2)\*gridSize]

\*/

private double distance(Player seeker) {

return Math.*hypot*(getX() - seeker.getX(), getY() - seeker.getY());

}

/\*\*

\* **@param** s

\* a Seeker `

\* **@return** 0, if the Seeker has found the hidden object. Returns a larger

\* int-valued "temperature" if the Seeker is farther away.

\*/

int revealTemperature(Seeker s) {

// compute a scaled distance. The range of r is 0.0 to 1.414

double r = distance(s) / (double) getGame().getGridSize();

System.***out***.print(" " + getName() + " says to " + s.getName() + ", \"");

if (r == 0.0) {

System.***out***.println("Huckle buckle beanstalk!\"");

return 0;

} else if (r <= 0.25) {

System.***out***.println("You're boiling!\"");

return 1;

} else if (r <= 0.35) {

System.***out***.println("You're hot.\"");

return 2;

} else if (r <= 0.5) {

System.***out***.println("You're warm.\"");

return 3;

} else if (r <= 0.7) {

System.***out***.println("You're cool.\"");

return 4;

} else if (r <= 1.0) {

System.***out***.println("You're cold.\"");

return 5;

} else {

System.***out***.println("freezing!\"");

return 6;

}

}

}

HuckleBuckle.java

package hucklebuckle;

/\*\*

\* A non-interactive console application which simulates a children's game.

\* <p>

\* Optional command-line parameter: gridsize, the size of the playing field, an

\* integer between 1 and 40 inclusive.

\* <p>

\* When versioning this code, programmers should adjust the value of

\* HuckleBuckle.VERSION so that an accurate version number appears on the first

\* line of program output.

\*

\* **@author** Clark Thomborson

\*

\*/

public class HuckleBuckle {

public static final String ***VERSION*** = "1.0";

public static void main(String[] args) {

int gridSize = 5; // gridSize == 5, if there are no command-line args

if (args.length > 0) {

try {

gridSize = Integer.*parseInt*(args[0]);

} catch (NumberFormatException e) {

System.***err***

.println("Error: first arg (gridSize) must be an integer. ");

System.*exit*(1);

}

if (gridSize < 1 || gridSize > 40) {

System.***err***

.println("Error: first arg (gridSize) must be in the range 1..40. ");

System.*exit*(1);

}

}

if (args.length > 1) {

System.***err***.println("Warning: too many args. ");

}

}

}

Player.java

package hucklebuckle;

class Player {

private String myName;

private Game myGame;

private int x; // constraint: 0 <= x < HuckleBuckle.GridSize

private int y; // constraint: 0 <= y < HuckleBuckle.GridSize

Player(String name, Game game) {

myName = name;

myGame = game;

setX(0); // all players start at (0,0)

setY(0);

}

String getName() {

return myName;

}

Game getGame() {

return myGame;

}

int getX() {

return x;

}

void setX(int px) {

x = (int) px; // **TODO**: add bounds-checking logic

}

int getY() {

return y;

}

void setY(int py) {

y = (int) py; // **TODO**: add bounds-checking logic

}

}

Seeker.java

package hucklebuckle;

class Seeker extends Player {

private Hider myHider;

private int distanceMoved; // updated after every change of position

Seeker(String name, Game game) {

super(name, game);

distanceMoved = 0;

}

void sayHelloTo(Hider h) {

myHider = h;

System.***out***.println(getName() + " says, \"Hi, " + h.getName() + ", I'm "

+ getName() + ". Glad to meet you!\"");

}

void pleasePlayWith(Hider hider, Game game) {

hider.introduceYourself();

hider.hide();

}

private void reportLocation() {

System.***out***.println(getName() + " says, \"I'm at " + getX() + ", "

+ getY() + ".\"");

}

void seek() {

reportLocation(); // I report my initial location

boolean moveRight = true; // I move left-to-right on even-numbered rows

for (int y = 0; y < getGame().getGridSize(); y++) {

if (getY() != y) {

setY(y); // I move to row y, if I'm not already there

distanceMoved++;

reportLocation(); // I announce my location after every move

}

for (int cx = 0; cx < getGame().getGridSize(); cx++) {

// cx: counts the number of times I have moved on row y

int x = (moveRight ? cx : getGame().getGridSize() - cx - 1);

if (getX() != x) {

setX(x);

distanceMoved++;

reportLocation(); // I announce my location after every move

}

if (myHider.revealTemperature(this) == 0) { // Did I find it?

System.***out***.println(getName()

+ " says, \"That was fun! I walked "

+ distanceMoved + " steps before I found it.\"");

return; // I stop when I find the hidden object

}

}

// I move in the opposite direction on each row of the grid

moveRight = !moveRight;

// My movement pattern is sometimes called the snake-like row-major

// ordering. See http://en.wikipedia.org/wiki/Boustrophedon.

}

System.***out***.println(getName() + " says, \"I'm giving up. I took "

+ distanceMoved + "steps before quitting.\"");

}

}