PRACTICA 3

Introducción a la Programación Orientada

A Objetos con Java

Alejandro Antonio Martín Almansa

Mario Garcia Roque

Apartado 4:

En el apartado 4 de esta práctica realizamos pruebas para probar las clases implementadas, es decir Posada, Camino y Explorador (El código de estas se mostrará en el Apartado 6 junto con las mejoras y ampliaciones).

EjemploDeUsoExploradoresBasico:

El código es la prueba que se nos da en el enunciado.

/\*\*

\* Prueba de las clases Explorador, POsada y Camino

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **class** EjemploDeUsoExploradoresBasicos {

// Ejemplo del Enunciado de la Practica

**public** **static** **void** main(String[] args) {

Posada solana = **new** Posada("Solana", 1);

Posada romeral = **new** Posada("Romeral", 5);

Posada tomelloso = **new** Posada("Tomelloso"); // por defecto energía recuperada 2

Explorador sancho = **new** Explorador("Sancho", 50, solana);

solana.addCamino(**new** Camino(solana, romeral, 68));

solana.addCamino(**new** Camino(solana, tomelloso, 33));

System.***out***.println(sancho);

sancho.recorre(romeral, tomelloso); // irá directo a tomelloso sin pasar por romeral

System.***out***.println(sancho); // energía 19 = 50 - 33 + 2

tomelloso.addCamino(**new** Camino(tomelloso, romeral, 11));

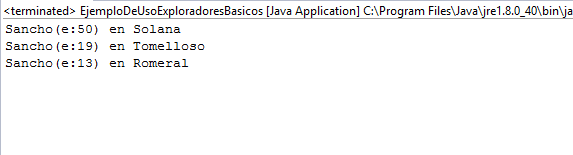
sancho.recorre(tomelloso.getCamino(romeral));

System.***out***.println(sancho); // en Romeral con energía 13 = 19 - 11 + 5

}

}

Salida: Imprime por pantalla lo que deseamos:



EjemploDeUsoExploradores1:

El código prueba al explorador para atravesar caminos, pero no tiene energía suficiente, es decir se quedará en solana siempre con la misma energía:

/\*\*

\* Prueba de las clases Explorador, POsada y Camino

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **class** EjemploDeUsoExploradores1 {

//Ejemplo que no tiene energia para ir a ninguna posada,

// y se queda siempre en la actual (solana)

**public** **static** **void** main(String[] args) {

Posada solana = **new** Posada("Solana", 1);

Posada romeral = **new** Posada("Romeral", 5);

Posada tomelloso = **new** Posada("Tomelloso"); // por defecto energia recuperada 2

Explorador sancho = **new** Explorador("Sancho", 30, solana);

solana.addCamino(**new** Camino(solana, romeral, 68));

solana.addCamino(**new** Camino(solana, tomelloso, 33));

System.***out***.println(sancho);

sancho.recorre(romeral, tomelloso); // No recorre ningun camino

System.***out***.println(sancho); // Se encuentra en solana

tomelloso.addCamino(**new** Camino(tomelloso, romeral, 11));

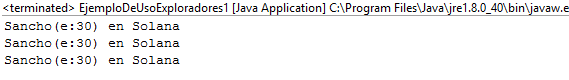
sancho.recorre(tomelloso.getCamino(romeral)); //No recorre camino

System.***out***.println(sancho); // Se encuentra en solana

}

}

Salida: La salida es la esperada:



EjemploDeUsoExploradores2:

El código prueba al explorador para atravesar caminos, que esta vez puede atravesarlos porque existe conexión y tiene energía suficiente:

/\*\*

\* Prueba de las clases Explorador, POsada y Camino

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **class** EjemploDeUsoExploradores2 {

//Ejemplo que puede recorrer los dos caminos por energia suficiente

**public** **static** **void** main(String[] args) {

Posada solana = **new** Posada("Solana", 1);

Posada romeral = **new** Posada("Romeral", 5);

Posada tomelloso = **new** Posada("Tomelloso"); // por defecto energia recuperada 2

Explorador sancho = **new** Explorador("Sancho", 80, solana);

solana.addCamino(**new** Camino(solana, romeral, 68));

solana.addCamino(**new** Camino(solana, tomelloso, 33));

romeral.addCamino(**new** Camino(romeral, tomelloso, 11));

System.***out***.println(sancho);

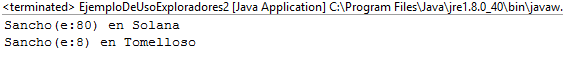
sancho.recorre(romeral, tomelloso); // ira a romeral

System.***out***.println(sancho); // energia 8 = 80 - 68 + 5 - 11 + 2

}

}

Salida: La esperada, es decir en tomelloso:



EjemploDeUsoExploradores3:

El código prueba al explorador para atravesar caminos, tiene energía suficiente, pero no puede atravesarlos porque no existen conexiones entre los caminos:

/\*\*

\* Prueba de las clases Explorador, POsada y Camino

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **class** EjemploDeUsoExploradores3 {

//Ejemplo energia suficiente pero no existen caminos para recorrer

**public** **static** **void** main(String[] args) {

Posada solana = **new** Posada("Solana", 1);

Posada romeral = **new** Posada("Romeral", 5);

Posada tomelloso = **new** Posada("Tomelloso"); // por defecto energia recuperada 2

Explorador sancho = **new** Explorador("Sancho", 50, solana);

System.***out***.println(sancho);

sancho.recorre(romeral, tomelloso); // No recorre ninguno porque no hay caminos

System.***out***.println(sancho); // En solana

tomelloso.addCamino(**new** Camino(tomelloso, romeral, 11));

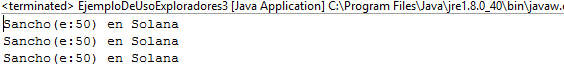
sancho.recorre(tomelloso.getCamino(romeral));

System.***out***.println(sancho); // En solana con la energia inicial

}

}

Salida: La esperada, en la posada inicial con toda la energía:



EjemploDeUsoExploradores4:

El código prueba al explorador para atravesar caminos, pudiendo recorrer todos. En esta prueba se usa la función cambiarDestino:

/\*\*

\* Prueba de las clases Explorador, POsada y Camino

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **class** EjemploDeUsoExploradores4 {

// Ejemplo usando funcion cambiarDestino, puediendo recorrer los caminos

**public** **static** **void** main(String[] args) {

Posada solana = **new** Posada("Solana", 1);

Posada romeral = **new** Posada("Romeral", 5);

Posada tomelloso = **new** Posada("Tomelloso");

Camino corto = **new** Camino(solana, romeral, 68);

Explorador sancho = **new** Explorador("Sancho", 100, solana);

solana.addCamino(corto);

corto.cambiarDestino(tomelloso, 33); // Corto= Origen: Solana,

// Destino: Tomelloso

System.***out***.println(sancho);

sancho.recorre(tomelloso);

System.***out***.println(sancho); // En tomelloso

tomelloso.addCamino(**new** Camino(tomelloso, romeral, 11));

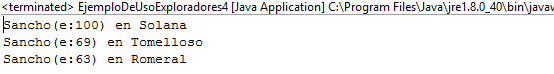
sancho.recorre(tomelloso.getCamino(romeral));

System.***out***.println(sancho); // En romeral

}

}

Salida: La esperada. El explorador en la posada romeral, después de pasar por tomelloso:



Apartado 5:

En este apartado realizamos la clase Simulación para crear caminos, posadas y exploradores desde ficheros. Desde estos ficheros debemos leer los datos por filas, y separando estas filas con la función de java Split, que nos separa una línea en cadenas pasándole por parámetro el separador.

Código de la clase Simulación:

/\*\*

\* Clase Simulacion

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**import** java.io.\*;

**import** java.util.\*;

**public** **class** Simulacion {

**private** ArrayList<Posada> posadas;

**private** ArrayList<Camino> caminos;

**private** Explorador explorador;

/\*\*

\* Constructor de la clase Simulacion

\* **@param** p Fichero de posadas

\* **@param** c Fichero de Caminos

\* **@param** e Fichero de Explorador

\* **@throws** IOException si hay fallo en los ficheros

\*/

**public** Simulacion(String p, String c, String e) **throws** IOException {

**this**.posadas = **new** ArrayList<Posada>();

String linea, nombre\_posada, energia;

**int** ener\_rec;

String[] pvalores;

FileInputStream fichero\_pos = **new** FileInputStream(p);

InputStreamReader preader = **new** InputStreamReader(fichero\_pos);

BufferedReader pbuffer = **new** BufferedReader(preader);

**while** ((linea = pbuffer.readLine()) != **null**) {

pvalores = linea.split(" ");

nombre\_posada = pvalores[0];

energia = pvalores[1];

ener\_rec = Integer.*parseInt*(energia);

posadas.add(**new** Posada(nombre\_posada, ener\_rec));

}

fichero\_pos.close();

**this**.caminos = **new** ArrayList<Camino>();

String clinea, nombre\_pos\_origen, nombre\_pos\_destino, energia\_camino;

**int** ener\_camino;

Posada pos\_origen = **null**, pos\_destino = **null**;

String[] cvalores;

FileInputStream fichero\_cam = **new** FileInputStream(c);

InputStreamReader creader = **new** InputStreamReader(fichero\_cam);

BufferedReader cbuffer = **new** BufferedReader(creader);

**while** ((clinea = cbuffer.readLine()) != **null**) {

cvalores = clinea.split(" ");

nombre\_pos\_origen = cvalores[0];

nombre\_pos\_destino = cvalores[1];

energia\_camino = cvalores[2];

ener\_camino = Integer.*parseInt*(energia\_camino);

**for** (Posada pos : posadas) {

**if** (pos.getNombre().compareToIgnoreCase(nombre\_pos\_origen)==0) {

pos\_origen = pos;

}

**if** (pos.getNombre().compareToIgnoreCase(nombre\_pos\_destino)==0) {

pos\_destino = pos;

}

}

caminos.add(**new** Camino(pos\_origen, pos\_destino, ener\_camino));

pos\_origen.addCamino(caminos.get(caminos.size()-1));

}

fichero\_cam.close();

String elinea, nombre\_exp, ener, posicion, pos\_dest\_exp;

**int** ener\_exp;

String[] evalores;

FileInputStream fichero\_exp = **new** FileInputStream(e);

InputStreamReader ereader = **new** InputStreamReader(fichero\_exp);

BufferedReader ebuffer = **new** BufferedReader(ereader);

**while** ((elinea = ebuffer.readLine()) != **null**) {

**if** (elinea.length() != 1) {

evalores = elinea.split(" ");

nombre\_exp = evalores[0];

ener = evalores[1];

ener\_exp = Integer.*parseInt*(ener);

posicion = evalores[2];

**for** (Posada p1 : posadas) {

**if** (p1.getNombre().compareToIgnoreCase(posicion)==0) {

explorador = **new** Explorador(nombre\_exp, ener\_exp, p1);

}

}

} **else** {

pos\_dest\_exp = elinea;

**for** (Posada p2 : posadas) {

**if** (p2.getNombre().compareToIgnoreCase(pos\_dest\_exp)==0) {

explorador.recorre(explorador.getPos().getCamino(p2));

System.***out***.println(explorador);

}

}

}

}

fichero\_exp.close();

**return**;

}

}

Para probar esta clase utilizamos los ficheros POSADAS.txt, CAMINOS.txt y EXPLORADOR.txt, que se encuentran en la carpeta raíz del programa, y el código de ejemplo que se nos da:

EjemploDeUsoSimulacion:

/\*\*

\* Prueba la clase Simulacion

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**import** java.io.IOException;

**public** **class** EjemploDeUsoSimulacion {

**public** **static** **void** main(String[] args) {

**try** {

Simulacion s;

s = **new** Simulacion("POSADAS.txt", "CAMINOS.txt", "EXPLORADOR.txt");

s = **new** Simulacion("POSADAS.txt", "CAMINOS.txt", "EXPLORADOR2.txt");

} **catch** (IOException e) {

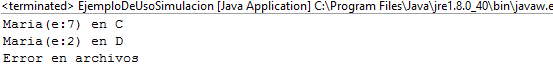
System.***out***.println("Error en archivos");

}

}

}

Salida: La esperada. Es decir recorre los dos primeros caminos y luego da un error porque no existe el fichero EXPLORADOR2.txt:



Apartado 6:

En el apartado 6 modificamos las clases de los apartados 1, 2 y 3 actualizando clases, añadiéndoles métodos y creando clases nuevas.

Clase Camino:

/\*\*

\* Clase Camino

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**import** p3.Posada;

**public** **class** Camino {

**private** Posada origen;

**private** Posada destino;

**private** **int** energia;

**private** **static** **int** *acumulado*;

/\*\*

\* Constructor de la clase Camino.

\* **@param** origen La posada en la que empieza el camino.

\* **@param** destino La en la que termina el camino.

\* **@param** energia La energia que consume recorrer el camino.

\*/

**public** Camino(Posada origen,Posada destino ,**int** energia) {

**this**.origen=origen;

**this**.destino=destino;

**this**.energia=energia;

}

/\*\*

\* Retorna el valor de acumulada.

\* **@return** Un entero con el valor de acumulado.

\*/

**public** **static** **int** getAcumulado() {

**return** *acumulado*;

}

/\*\*

\* Retorna la posada origen.

\* **@return** Un objeto Posada con el vlaor de la posada origen.

\*/

**public** Posada getOrigen() {

**return** origen;

}

/\*\*

\* Retorna la posada destino.

\* **@return** Un objeto Posada con el valor de la posada destino .

\*/

**public** Posada getDestino() {

**return** destino;

}

/\*\*

\* Retorna la energia que consume recorrer el camino.

\* **@return** Un entero con el coste de recorrer el camino.

\*/

**public** Integer getEnergia() {

**return** energia;

}

/\*\*

\* Cambia la posada destino y la energia que cuesta recorrer el camino.

\* **@param** dst Un objeto Posada con la nueva posada destino.

\* **@param** enrg Un entero con el valor de recorrer el camino.

\*/

**public** **void** cambiarDestino(Posada dst,**int** enrg){

**this**.destino= dst;

**if**(enrg<1){

**this**.energia=1;

}

**else**{

**this**.energia=enrg;

}

}

/\*\*

\* Devuelve el coste especial.

\* **@return** Entero con el valor del coste especial.

\*/

**public** **int** costeEspecial(){

**return** 0;

}

/\*\*

\* Devuelve el coste real.

\* **@return** Entero con el valor del coste real.

\*/

**public** **int** costeReal(){

**return** **this**.costeEspecial()+**this**.energia;

}

/\*\*

\* Devuelve si el camino es una trampa.

\* **@return** Un boleano con el valor de si el camino es una trampa.

\*/

**public** **boolean** esTrampa (){

**return** **false**;

}

/\*\*

\* Devuelve los valores del objeto como una cadena.

\* **@return** String con el valor del objeto convertido a cadena.

\*/

**public** String toString() {

**return** "("+**this**.origen.getNombre()+"--"+**this**.energia+"-->"+**this**.origen.getNombre();

}

}

Clase Posada:

/\*\*

\* Clase Posada

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**import** p3.Camino;

**import** java.util.\*;

**public** **class** Posada {

**private** String nombre;

**private** **int** enerRec;

**private** ArrayList <Camino>salidas;

**public** **static** **final** **int** ***ENER\_REC*** = 2;

**private** Luz luz;

/\*\*

\* Constructor de la clase Posada.

\* **@param** nombre Nombre de la posada.

\*/

**public** Posada (String nombre) {

**this**(nombre,***ENER\_REC*** );

}

/\*\*

\* Constructor de la clase Posada.

\* **@param** nombre Nombre de la posada.

\* **@param** ener Energia que se recupera al entrar en una posada.

\*/

**public** Posada (String nombre, **int** ener) {

**this**.nombre = nombre;

**this**.enerRec = ener;

**this**.salidas = **new** ArrayList<Camino>();

**this**.luz = Luz.***BLANCA***;

}

/\*\*

\* Constructor de la clase Posada.

\* **@param** nombre Nombre de la posada.

\* **@param** ener Energia que se recupera al entrar en una posada.

\* **@param** luz El tipo de luz que hay en la posada.

\*/

**public** Posada (String nombre, **int** ener, Luz luz) {

**this**(nombre,ener);

**this**.luz = luz;

}

/\*\*

\* Devuelve el nombre.

\* **@return** Devuelve un String con el nombre.

\*/

**public** String getNombre() {

**return** **this**.nombre;

}

/\*\*

\* Devuelve la energia de la posada.

\* **@return** Entero con la energia ecuperada.

\*/

**public** **int** getEnerRec() {

**return** **this**.enerRec;

}

/\*\*

\* Dado un entero devuelve un camino de la lista de caminos que

\* se corresponde con el elemento de la lista.

\* **@param** c Entero con el camino que se quiere acceder.

\* **@return** El camino indicado.

\*/

**public** Camino getCamino(**int** c) {

**return** **this**.salidas.get(c);

}

/\*\*

\* Devuelve el numero de caminos.

\* **@return** Un entero con el numero de caminos.

\*/

**public** **int** getNumCaminos() {

**return** **this**.salidas.size();

}

/\*\*

\* Dada una posada devuelve el camino que une a la posada actual

\* y la posada que nos pasan.

\* **@param** p Posada que nos pasan.

\* **@return** Camino que une ambas posadas.

\*/

**public** Camino getCamino (Posada p) {

**for** (Camino c: salidas) {

**if** (c.getDestino().nombre==p.nombre)

**return** c;

}

**return** **null**;

}

/\*\*

\* Añade una camino a la lista de salidas.

\* **@param** c Nuevo camino.

\* **@return** Boleano que indica si se ha podido añadir el camino.

\*/

**public** **boolean** addCamino (Camino c) {

**if**(**this**.salidas.contains(c))

**return** **false**;

**if** (**this**.salidas.add(c)==**true**)

**return** **true**;

**return** **false**;

}

/\*\*

\* Dada una luz cambia la luz de la posada.

\* **@param** luz Nueva luz de la posada

\*/

**void** cambiarLuz (Luz luz) {

**this**.luz = luz;

}

/\*\*

\* Devuelve la luz de la posada

\* **@return** Luz de la posada

\*/

**public** Luz getLuz() {

**return** luz;

}

/\*\*

\* Transforma el objeto Posada en un String

\* **@return** String con el objeto

\*/

**public** String toString() {

**return** **this**.nombre+"("+**this**.enerRec+") ["+**this**.salidas.toString()+"]";

}

}

Clase Explorador:

/\*\*

\* Clase Explorador

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **class** Explorador {

**private** String nombre;

**private** **int** energia;

**private** Posada pos;

/\*\*

\* Constructor de la clase Explorador.

\* **@param** nombre Nombre del explorador.

\* **@param** energia Energia del explorador.

\* **@param** iniPos Posada en la que se encuentra el explorador.

\*/

**public** Explorador(String nombre, **int** energia, Posada iniPos) {

**super**();

**this**.nombre = nombre;

**this**.energia = energia;

**this**.pos = iniPos;

}

/\*\*

\* Devuelve el nombre del explorador.

\* **@return** String con el nombre del explorador.

\*/

**public** String getNombre() {

**return** nombre;

}

/\*\*

\* Devuelve la energia del explorador.

\* **@return** Entero con la energia del explorador.

\*/

**public** **int** getEnergia() {

**return** energia;

}

/\*\*

\* Devuelve la porsada donde se encuentra el explorador.

\* **@return** Objeto con la posada donde se encuentra el explorador.

\*/

**public** Posada getPos() {

**return** pos;

}

/\*\*

\* Dado un camino devuelve si el explorador

\* puede recorrer el camino o no.

\* **@param** c Camino a recorrer.

\* **@return** Un valor boleano con el resultado de la operacion.

\*/

**public** **boolean** recorre(Camino c) {

**if** (c == **null** || c.getOrigen() != **this**.pos)

**return** **false**;

Posada pos\_dest = c.getDestino();

**if** (**this**.puedeRecorrerCamino(c) && **this**.puedeAlojarseEn(pos\_dest)) {

**this**.pos = pos\_dest;

**this**.energia -= c.costeReal();

**this**.energia += pos\_dest.getEnerRec();

**return** **true**;

}

**return** **false**;

}

/\*\*

\* Dada una lista de Posadas recorre caminos hasta llegar a las

\* posadas indicadas, si puede.

\* Si no se puede recorrer un camino la funcion devolvera FALSE pero

\* recorrera todos los caminos intentando llegar a la posada final.

\* **@param** ps Una lista con las posadas que queremos atravesar.

\* **@return** Un booleano que nos indica si se ha podido recorrer todas las psadas.

\*/

**public** **boolean** recorre(Posada... ps) {

Boolean ret = **true**;

**for** (Posada p : ps) {

**if** (**this**.recorre(**this**.pos.getCamino(p)) == **false**) {

ret = **false**;

}

}

**return** ret;

}

/\*\*

\* Metodo que dado un camino comprueba si se puede recorrer el camino o no.

\* **@param** c Camino a recorrer

\* **@return** Boleano que nos indica si se puede recorrer el camino.

\*/

**public** **boolean** puedeRecorrerCamino(Camino c) {

**if** (c == **null**)

**return** **false**;

**if** (**this**.energia > c.costeReal())

**return** **true**;

**return** **false**;

}

/\*\*

\* Metodo que indica si el explorador puede alojarse en una posada.

\* **@param** p Posada en la que nos podiramos alojar.

\* **@return** Boleano con el valor de si nos podemos alojar o no.

\*/

**public** **boolean** puedeAlojarseEn(Posada p) {

**return** **true**;

}

/\*\*

\* Metodo que convierte el objeto a formato String.

\* **@return** String con la informacion del explorador.

\*/

**public** String toString() {

**return** **this**.nombre + "(e:" + **this**.energia + ") en "

+ **this**.pos.getNombre();

}

}

Clase Trampa:

/\*\*

\* Clase Trampa

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**import** p3.Posada;

**public** **class** Trampa **extends** Camino {

**private** **int** factor\_coste\_extra;

**private** **int** prob\_ret\_obligado;

/\*\*

\* Constructor de la clase Trampa

\*

\* **@param** origen

\* Posada origen.

\* **@param** destino

\* Posada Destino.

\* **@param** energia

\* Energia que se pierde al recorrer el camino.

\* **@param** factor\_c\_ext

\* Factor extra que cuesta atravesar la trampa

\* **@param** prob\_r\_oblig

\* Probabilidad de caer en la trampa (del 1 al 10)

\*/

**public** Trampa(Posada origen, Posada destino, **int** energia, **int** factor\_c\_ext,

**int** prob\_r\_oblig) {

**super**(origen, destino, energia);

**this**.factor\_coste\_extra = factor\_c\_ext;

**this**.prob\_ret\_obligado = prob\_r\_oblig;

}

/\*\*

\* Coste especial de recorrer el camino.

\*

\* **@return** Entero con el coste de recorrer el camino.

\*/

@Override

**public** **int** costeEspecial() {

**return** **this**.factor\_coste\_extra \* **this**.getEnergia();

}

/\*\*

\* Retorna el factor coste extra.

\*

\* **@return** Un int con el valor del factor.

\*/

**public** **int** getFactor\_coste\_extra() {

**return** factor\_coste\_extra;

}

/\*\*

\* Retorna la probabilidad de retorno obligatorio.

\*

\* **@return** Un int con la probabilidad.

\*/

**public** **int** getProb\_ret\_obligado() {

**return** prob\_ret\_obligado;

}

/\*\*

\* Retorna la posada de destino

\*

\* **@return** Una posada con el valor del destino de la trampa si la

\* probabilidad de retorno obligatorio es mayor que el valor

\* aleatorio y con la posada de origen si esa probabilidad es menor

\* que el valor aleatorio.

\*/

@Override

**public** Posada getDestino() {

**int** valorEntero = (**int**) Math.*floor*(Math.*random*() \* (10 - 0 + 1) + 0);

**if** (valorEntero <= **this**.getProb\_ret\_obligado()) {

**return** **this**.getOrigen();

}

**return** **super**.getDestino();

}

/\*\*

\* Metodo que devuelve si es una trampa.

\*

\* **@return** Devuelve TRUE.

\*/

@Override

**public** **boolean** esTrampa() {

**return** **true**;

}

/\*\*

\* Devuelve los valores del objeto como una cadena.

\*

\* **@return** String con el valor del objeto convertido a cadena.

\*/

**public** String toString() {

**return** "Trampa: (" + **this**.getOrigen().getNombre() + "--"

+ **this**.getEnergia() + "-->" + **this**.getOrigen().getNombre();

}

}

Clase Enumerado Luz:

/\*\*

\* Clase Enumerado Luz

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **enum** Luz {

***DIABOLICA***(0),

***NEGRA***(1),

***TENEBROSA***(2),

***GRIS***(3),

***CLARA***(4),

***BLANCA***(5),

***DIVINA***(6);

**private** **int** numero;

/\*\*

\* Constructor de la clase Luz

\* **@param** numero Valor numerico de la luz

\*/

Luz(**int** numero) {

**this**.numero = numero;

}

/\*\*

\* Retorna el valor numerico de la luz

\* **@return** Entero con el valor de la luz

\*/

**public** **int** getNumero() {

**return** numero;

}

}

Clase Mago:

/\*\*

\* Clase Mago

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **class** Mago **extends** Explorador {

**private** **int** poder;

**public** Mago(String nombre, **int** energia, Posada iniPos, **int** poder) {

**super**(nombre, energia, iniPos);

**this**.poder = poder;

}

**public** **boolean** puedeRecorrerCamino(Camino c) {

**if** (c == **null** || c.esTrampa()==**true**)

**return** **false**;

**if** (**this**.getEnergia() > c.costeReal())

**return** **true**;

**return** **false**;

}

**public** **int** getPoder() {

**return** poder;

}

**public** String toString() {

**return** "Mago: " + **this**.getNombre() + " (e:" + **this**.getEnergia()

+ ") en " + **this**.getPos().getNombre();

}

}

Clase Hechicero:

/\*\*

\* Clase Hechicero

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **class** Hechicero **extends** Mago {

/\*\*

\* Constructor de la clase Hechicero

\* **@param** nombre Nombre del Hechicero

\* **@param** energia Energia del Hechicero

\* **@param** iniPos Posicion inicial del Hechicero

\* **@param** poder Poder del Hechicero

\*/

**public** Hechicero(String nombre, **int** energia, Posada iniPos, **int** poder) {

**super**(nombre, energia, iniPos, poder);

}

/\*\*

\* Metodo que indica si el hechicero puede alojarse en una posada.

\* **@param** p Posada en la que nos podiramos alojar.

\* **@return** Boleano con el valor de si nos podemos alojar o no.

\*/

@Override

**public** **boolean** puedeAlojarseEn(Posada p) {

**if** (p.getLuz().getNumero() <= 2

|| (p.getLuz().getNumero() > 2 && **this**.getPoder() >= p.getLuz()

.getNumero())) {

**return** **true**;

}

**return** **false**;

}

/\*\*

\* Metodo que convierte el objeto a formato String.

\* **@return** String con la informacion del explorador.

\*/

**public** String toString() {

**return** "Hechicero: " + **this**.getNombre() + " (e:" + **this**.getEnergia()

+ ") en " + **this**.getPos().getNombre();

}

}

Clase Hada:

/\*\*

\* Clase Hada

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **class** Hada **extends** Mago {

/\*\*

\* Constructor de la clase Hada

\* **@param** nombre Nombre de la Hada

\* **@param** energia Energia de la Hada

\* **@param** iniPos POsicion inicial de la Hada

\* **@param** poder Poder de la hada

\*/

**public** Hada(String nombre, **int** energia, Posada iniPos, **int** poder) {

**super**(nombre, energia, iniPos, poder);

}

/\*\*

\* Metodo que indica si la hada puede alojarse en una posada.

\* **@param** p Posada en la que nos podiramos alojar.

\* **@return** Boleano con el valor de si nos podemos alojar o no.

\*/

@Override

**public** **boolean** puedeAlojarseEn(Posada p) {

**if** (p.getLuz().getNumero() > 3 ) {

**return** **true**;

}

**return** **false**;

}

/\*\*

\* Metodo que convierte el objeto a formato String.

\* **@return** String con la informacion del explorador.

\*/

**public** String toString() {

**return** "Hada: " + **this**.getNombre() + " (e:" + **this**.getEnergia()

+ ") en " + **this**.getPos().getNombre();

}

}

Para probar estas clases implementadas en el ejercicio 6, es decir las de Mago, Hada, Hechicero, Luz y Trampa hacemos las 2 siguientes pruebas:

EjemploDeUsoTrampa:

El código prueba si el explorador recorre las trampas.

/\*\*

\* Prueba de las clases Explorador, Posada, Camino y Trampa

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **class** EjemploDeUsoTrampa {

**public** **static** **void** main(String[] args) {

Posada solana = **new** Posada("Solana", 1);

Posada romeral = **new** Posada("Romeral", 5);

Posada tomelloso = **new** Posada("Tomelloso"); // por defecto enerRec = 2

Explorador sancho = **new** Explorador("Sancho", 100, solana);

solana.addCamino(**new** Trampa(solana, romeral, 20, 1, 3));

romeral.addCamino(**new** Camino(romeral, tomelloso, 11));

System.***out***.println(sancho);

sancho.recorre(romeral); // ira a romeral si la trampa no se activa

System.***out***.println(sancho); // si va a romeral: 65 = 100 - (20\*1 + 20)+ 5

sancho.recorre(romeral.getCamino(tomelloso));

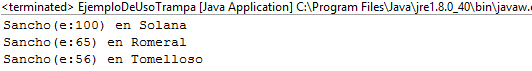
System.***out***.println(sancho);

}

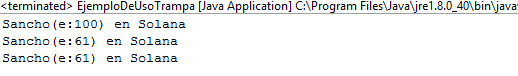
}

Salida: La esperada. Dependiendo si recorre la trampa o vuelve a la posada de origen tendremos dos salidas distintas:

Salida si pasa la trampa:



Salida si no pasa la trampa:



EjemploDeUsoMago:

Se crean 3 personajes, un explorador, un hechicero y un hada. Para estos tres personajes crearemos caminos, trampas y posadas y tendrán que recorrer estos según le pasen unos caminos u otros:

/\*\*

\* Prueba de las clases Mago, Hechicero y Hada

\* **@author** Alejandro Antonio Martin Almansa

\* **@author** Mario Garcia Roque

\*

\*/

**package** p3;

**public** **class** EjemploDeUsoMago {

// Se crean 3 personajes que irán recorriendo los caminos y trampas según su

// energía y sus permisos.

**public** **static** **void** main(String[] args) {

Posada solana = **new** Posada("Solana", 1); // luz blanca por defecto

Posada romeral = **new** Posada("Romeral", 5, Luz.***TENEBROSA***);

Posada tomelloso = **new** Posada("Tomelloso", 5, Luz.***NEGRA***);

Posada pisador = **new** Posada("Nevadalia", 10, Luz.***BLANCA***);

Posada comarca = **new** Posada("Toboso", 15, Luz.***DIVINA***);

Explorador sancho = **new** Explorador("Sancho", 50, solana);

Explorador gandalf = **new** Hechicero("Gandalf", 100, pisador, 5);

Explorador arwen = **new** Hada("Arwen", 150, romeral, 20);

solana.addCamino(**new** Trampa(solana, romeral, 15, 1, 3));

romeral.addCamino(**new** Camino(romeral, pisador, 5));

romeral.addCamino(**new** Camino(romeral, solana, 10));

tomelloso.addCamino(**new** Camino(tomelloso, comarca, 10));

pisador.addCamino(**new** Camino(pisador, tomelloso, 10));

pisador.addCamino(**new** Camino(pisador, solana, 10));

/\*\*

\* Va a poder ir a romeral si no se activa la trampa. Podrá ir a pisador

\* si llega a romeral. No podrá ir a solana porque no hay camino entre

\* pisador y solana.

\*/

System.***out***.println(sancho);

sancho.recorre(romeral, pisador, solana);

System.***out***.println(sancho);

/\*\*

\* Podra ir a tomelloso, pero no a comarca por poder insuficiente.

\*/

System.***out***.println(gandalf);

gandalf.recorre(tomelloso, comarca);

System.***out***.println(gandalf);

/\*\*

\* Podra ir a solana, pero no a romeral porque no pueden pasar por una

\* trampa.

\* \*/

System.***out***.println(arwen);

romeral.cambiarLuz(Luz.***DIVINA***);

arwen.recorre(solana, romeral);

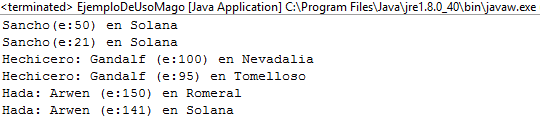
System.***out***.println(arwen);

}

}

Salida: La esperada. Dos casos para sancho (el explorador), uno si cae en la trampa y vuelve al origen y otro si pasa la trampa:

Si cae en la trampa:



Si no cae en la trampa:

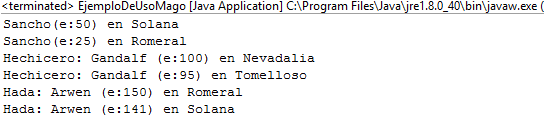


DIAGRAMA DE CLASES:

