Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 6

З дисципліни «Інженерія програмного забезпечення»

На тему « ШАБЛОНИ ПОВЕДІНКИ. ШАБЛОНИ STRATEGY, CHAIN OF RESPONSIBILITY, VISITOR»

ВИКОНАВ:

студент 2 курсу ФІОТ

групи ІО-81

Дудка Максим

Залікова - 8106

Варіант – 6

ПЕРЕВІРИВ:

Старший викладач

к.т.н., с.н.с.

Антонюк А.І.

Київ – 2019

ЗАВДАННЯ

Варіант 6.

Визначити специфікації класів для елементу ігрового поля (комірки) та самого простору. Забезпечити слабку зв'язаність елементів. Реалізувати децентралізований механізм сумісної зміни стану елементів.

ПРОГРАМНИЙ КОД

package com.lab111.labwork6;  
  
*/\*\*  
 \* Class which formed game elements  
 \** ***@author*** *Dudka Maxym  
 \*  
 \*/*public class ClientElement {  
 */\*\*  
 \** ***@param*** *args  
 \*/* private int posX;  
 private int posY;  
 private String name;  
  
 */\*\*  
 \*  
 \** ***@param*** *posX set x-position of element on field  
 \** ***@param*** *posY set y-position of element on field  
 \** ***@param*** *name set name of element  
 \*/* public ClientElement(int posX, int posY, String name){  
 this.posX=posX;  
 this.posY=posY;  
 this.name=name;  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *Get position of element on the field  
 \*/* public int getPosX(){  
 return this.posX;  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *Get position of element on the field  
 \*/* public int getPosY(){  
 return this.posY;  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *Get name of element  
 \*/* public String getName(){  
 return this.name;  
 }  
  
}

package com.lab111.labwork6;  
  
*/\*\*  
 \* Class, which implements interface Handler and send information about game element  
 \** ***@author*** *Dudka Maxym  
 \*  
 \*/*public class ConcreteHandler1 implements Handler{  
 */\*\*  
 \** ***@param*** *args  
 \*/* public String message;  
 ClientElement[] clientelements;  
 PlayField playfield;  
  
 */\*\*  
 \*  
 \** ***@param*** *clientelements - set array of game elements  
 \** ***@param*** *playfield - set play field  
 \*/* public ConcreteHandler1(ClientElement[] clientelements, PlayField playfield){  
 this.clientelements=clientelements;  
 this.playfield = playfield;  
 }  
  
 */\*\*  
 \** ***@return*** *message about existence game element in the play field  
 \*/* public void HandlerRequest(ClientElement el){  
 if (el.getPosX()<=playfield.getWidth() && el.getPosX()>=0 &&  
 el.getPosY()<=playfield.getHeight() && el.getPosY()>=0){  
 message = getPositiveMessage(el);  
 } else {  
 message = getNegativeMessage(el);  
 }  
 }  
 */\*\*  
 \*  
 \** ***@param*** *el game element  
 \** ***@return*** *message confirm existence of game element  
 \*/* public String getPositiveMessage(ClientElement el){  
 return "Your element " +el.getName()+" in field.";  
 }  
  
 */\*\*  
 \*  
 \** ***@param*** *el game element  
 \** ***@return*** *message don`t confirm existence  
 \*/* public String getNegativeMessage(ClientElement el){  
 return "Your elemnt "+el.getName()+" doesn`t in field.";  
 }  
  
}

package com.lab111.labwork6;  
  
*/\*\*  
 \* Class, which implements interface Handler, take message from ConcereteHandler1 and  
 \* send info about possible way directions.  
 \** ***@author*** *Dudka Maxym  
 \*  
 \*/*public class ConcreteHandler2 implements Handler{  
 */\*\*  
 \** ***@param*** *args  
 \*/* private String endMessage;  
 ConcreteHandler1 concretehandler1;  
 public int i=0;  
  
 */\*\*  
 \*  
 \** ***@param*** *concretehandler1 - set handler, that give info  
 \*/* public ConcreteHandler2 (ConcreteHandler1 concretehandler1){  
 this.concretehandler1=concretehandler1;  
 };  
 */\*\*  
 \* Suggest about existence of possible ways  
 \** ***@param*** *el - game element  
 \*/* @Override  
 public void HandlerRequest(ClientElement el) {  
 // *TODO Auto-generated method stub* if ((concretehandler1.message).equals(concretehandler1.getPositiveMessage(el))){  
 whichWayDirection(el);  
 } else {  
 endMessage=concretehandler1.getNegativeMessage(el);  
 }  
  
 }  
  
 */\*\*  
 \*  
 \** ***@param*** *el game element  
 \** ***@param*** *dir possible direction  
 \** ***@return*** *free possible ways  
 \*/* public String getDirectionMove(ClientElement el, String dir){  
 return concretehandler1.getPositiveMessage(el)+" "+dir+" is free. ";  
 }  
  
 */\*\*  
 \* Choose direction for move  
 \** ***@param*** *el game element  
 \*/* public void whichWayDirection(ClientElement el){  
 if (i==concretehandler1.clientelements.length-1){  
 endMessage=getDirectionMove(concretehandler1.clientelements[i],"Right direction");  
 return;  
 }  
 if ((el.getPosX()+1)!=concretehandler1.clientelements[i+1].getPosX()){  
 endMessage=getDirectionMove(el,"Right direction");  
 } else if ((el.getPosX()-1)!=concretehandler1.clientelements[i+1].getPosX()){  
 endMessage=getDirectionMove(el,"Left direction");  
 } else if ((el.getPosY()+1)!=concretehandler1.clientelements[i+1].getPosY()){  
 endMessage=getDirectionMove(el, "Up direction");  
 } else if ((concretehandler1.clientelements[i].getPosY()-1)!=concretehandler1.clientelements[i+1].getPosY()){  
 endMessage=getDirectionMove(el,"Down direction");  
 }  
 i++;  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *message about moving  
 \*/* public String getEndMessage(){  
 return endMessage;  
 }  
}

package com.lab111.labwork6;  
*/\*\*  
 \* Determines playfield, where game elements is stay.  
 \** ***@author*** *Dudka Maxym  
 \*  
 \*  
 \*  
 \*/*public class PlayField {  
 private int width;  
 private int height;  
  
 public PlayField(int width, int height){  
 this.width=width;  
 this.height=height;  
 }  
  
 */\*\*  
 \* Getter  
 \*/* public int getWidth(){  
 return width;  
 }  
 */\*\*  
 \*  
 \* Getter  
 \*/* public int getHeight(){  
 return height;  
 }  
  
}

package com.lab111.labwork6;  
  
*/\*\*  
 \* Interface for strategy  
 \** ***@author*** *Dudka Maxym  
 \*  
 \*/*public interface Strategy {  
 public String AlgorithmInterface();  
}

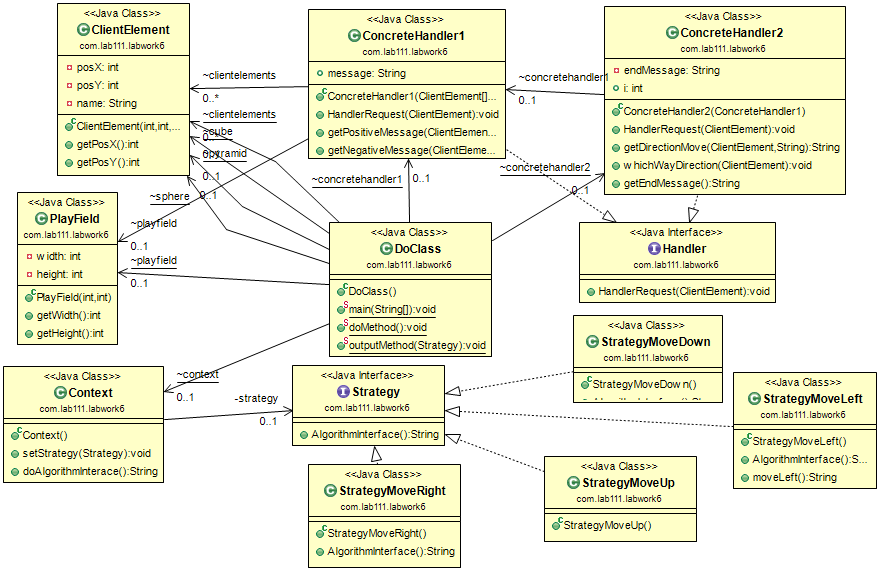
package com.lab111.labwork6;  
  
*/\*\*  
 \* Class, which implement interface Strategy and move element down  
 \** ***@author*** *Dudka Maxym  
 \*  
 \*/*public class StrategyMoveDown implements Strategy {  
  
 /\* (non-Javadoc)  
 \* @see com.lab111.labwork6.Strategy#AlgorithmInterace()  
 \*/  
 @Override  
 public String AlgorithmInterface() {  
 // *TODO Auto-generated method stub* return moveDown();  
 }  
  
 public String moveDown(){  
 return "Game element go down.";  
 }  
}

package com.lab111.labwork6;  
  
*/\*\*  
 \* Class, which implement interface Strategy and move element left  
 \** ***@author*** *Dudka Maxym  
 \*  
 \*/*public class StrategyMoveLeft implements Strategy{  
  
 @Override  
 public String AlgorithmInterface() {  
 // *TODO Auto-generated method stub* return moveLeft();  
 }  
 public String moveLeft(){  
 return "Game element go left.";  
 }  
}

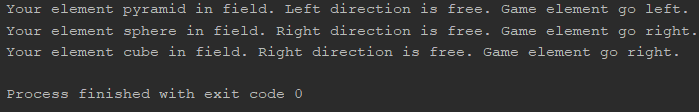
package com.lab111.labwork6;  
  
*/\*\*  
 \* Class, which implement interface Strategy and move element right  
 \** ***@author*** *Dudka Maxym  
 \*  
 \*/*public class StrategyMoveRight implements Strategy{  
  
 @Override  
 public String AlgorithmInterface() {  
 // *TODO Auto-generated method stub* return moveRight();  
 }  
  
 public String moveRight(){  
 return "Game element go right.";  
 }  
}

package com.lab111.labwork6;  
  
*/\*\*  
 \* Class, which implement interface Strategy and move element up  
 \** ***@author*** *Dudka Maxym  
 \*  
 \*/*public class StrategyMoveUp implements Strategy {  
  
 /\* (non-Javadoc)  
 \* @see com.lab111.labwork6.Strategy#AlgorithmInterace()  
 \*/  
 @Override  
 public String AlgorithmInterface() {  
 // *TODO Auto-generated method stub* return moveUp();  
 }  
  
 public String moveUp(){  
 return "Game element go up.";  
 }  
}

package com.lab111.labwork6;  
  
*/\*\*  
 \* Working class  
 \** ***@author*** *Dudka Maxym  
 \*  
 \*/*public class DoClass {  
  
 */\*\*  
 \** ***@param*** *args  
 \*/* static ClientElement *pyramid* = new ClientElement (26,78,"pyramid");  
 static ClientElement *sphere* = new ClientElement (27,58,"sphere");  
 static ClientElement *cube* = new ClientElement (43,63,"cube");  
 static ClientElement[] *clientelements* = {*pyramid*,*sphere*,*cube*};  
 static PlayField *playfield* = new PlayField(100,100);  
 static ConcreteHandler1 *concretehandler1* = new ConcreteHandler1 (*clientelements*, *playfield*);  
 static ConcreteHandler2 *concretehandler2* = new ConcreteHandler2 (*concretehandler1*);  
 static Context *context* = new Context();  
  
 public static void main(String[] args) {  
 // *TODO Auto-generated method stub  
 doMethod*();  
 }  
 */\*\*  
 \* Choose strategy for moving  
 \*/* public static void doMethod(){  
 for (ClientElement el: *clientelements*){  
 *concretehandler1*.HandlerRequest(el);  
 *concretehandler2*.HandlerRequest(el);  
 if (*concretehandler2*.getEndMessage().equals(*concretehandler2*.getDirectionMove(el,"Right direction"))){  
 *outputMethod*(new StrategyMoveRight());  
 } else if (*concretehandler2*.getEndMessage().equals(*concretehandler2*.getDirectionMove(el,"Left direction"))){  
 *outputMethod*(new StrategyMoveLeft());  
 } else if (*concretehandler2*.getEndMessage().equals(*concretehandler2*.getDirectionMove(el,"Up direction"))){  
 *outputMethod*(new StrategyMoveUp());  
 } else if (*concretehandler2*.getEndMessage().equals(*concretehandler2*.getDirectionMove(el,"Down direction"))){  
 *outputMethod*(new StrategyMoveDown());  
 } else if (*concretehandler1*.message.equals(*concretehandler1*.getNegativeMessage(el))){  
 System.*out*.println(*concretehandler1*.message);  
 }  
 }  
 }  
 */\*\*  
 \* Set strategy  
 \** ***@param*** *strategy strategy  
 \*/* public static void outputMethod(Strategy strategy){  
 *context*.setStrategy(strategy);  
 System.*out*.println(*concretehandler2*.getEndMessage()+*context*.doAlgorithmInterace());  
 }  
  
}



РЕЗУЛЬТАТИ ТЕСТУВАННЯ ПРОГРАМИ



ВИСНОВКИ

Було ознайомлено із шаблонами поведінки Strategy, Chain of responsibility та Visitor. Отримано навички із застосування шаблонів. Розроблено відповідна тестова програма. Результати успішної роботи тестової програми наведені вище підтверджують правильність обраних рішень.