Державний університет «Одеська Політехніка»
Інститут комп’ютерних систем
Кафедра інформаційних систем

Лабораторна робота №5
з дисципліни «Об’єктно-орієнтоване програмування»
Тема: «Повторне використання коду. Наслідування та композиція »

Виконав:
студент групи АІ-201
Мартинюк Д.В.

Прийняв:
доц. Годовиченко М.А.

Одеса 2021

Ход работы

1. Перегрузка методов
2. Цепочка наследования
3. Задание на простой полиморфизм
4. Модификация программы Draw

1. Перегрузка методов

В качестве основного методоа взят первый метод add и на его основе реализованы все остальные

```
public void add(final int hours final int minutes)
{

    // Check if input values are valid to add
    if(!isValidInput(hours minutes)){ return;}

    // Set new time
    this.currentTime.setHours(currentTime.getHours() + hours);
    this.currentTime.setMinutes(currentTime.getMinutes() + minutes);

}
public void add(final int minutes)
{
    add(0 minutes);
}
public void addTimeSpan(@NotNull TimeSpan otherTime)
{
   add(otherTime.getHours() otherTime.getMinutes());
}
```

2. Цепочка наследования

Геттеры и сеттеры в базовом классе

```
const string& GetName()const { return this->Name; }
const string& GetSurname()const { return this->Surname; }
const int GetAge()const { return this->Age; }

void SetName(const string name) { this->Name = name; }
void SetSurname(const string surname){ this->Surname = surname; }
void SetAge(const int age){ this->Age = age; }
```

Метод сделан виртуальным чтобы иметь возможность переопределения в будущем

```
virtual const string toString()
{
	return "Human " + toStringIntermediateSample();
}
```

Этот метод является повторяющимся шаблоном, поэтому я вынес его для удобного переопределения метода toString в классах наследниках

```
// General repeatable part of toString moved to separated method
const string toStringIntermediateSample() { return this->Surname + " " + this->Name + " ," + "age: " + to_string(this->Age); }
```

Данный метод(функция в данном случае) является промежуточным , в нем создается генератор случайных чисел и заполняем ними индексовый вектор, и на основе этого вектора индексов заполняется вектор smart pointers с нашими объектами.

```
vector<unique_ptr<Person>> GeneratePeopleVector()
{

	// An instance of an engine
	random_device random_device;

	// Specify the engine and distribution
	mt19937 engine{ random_device() };
	uniform_int_distribution<int> distribution{ 0, 2 };

	auto generator = [&distribution, &engine]() {   return
distribution(engine); };


	vector<int> indexes(10);
	generate(begin(indexes), end(indexes), generator);


	vector<unique_ptr<Person>>people;
	for (size_t i = 0; i < indexes.size(); i++)
	{

		switch (indexes[i])
		{
		case 0:
			people.push_back(make_unique<Person>(Person{}));
			break;
		case 1:
			people.push_back(make_unique<Student>(Student{}));
			break;
		case 2:
			people.push_back(make_unique<Lecturer>(Lecturer{}));
			break;
		}

	}


	return people;

}
```

3. Задание на простой полиморфизм

В методе generateShape вызывается отдельный метод который и будет создавать объекты изходя из сгенерированного переданого индекса дабы не захламлять метод.

```
 // Returns randomly generated object
private GameShape generateShape()
{
    int random = new Random().nextInt(3);

    return randomShape(new GameShape()  random);
}

// Returns shape depending on generated index
private GameShape randomShape(GameShape shape  int random)
{

    switch (random)
    {

        case 0:
            shape = new Rock();
            return shape;

        case 1:
            shape = new Paper();
            return shape;

        case 2:
            shape = new Scissors();
            return shape;

    }

    return shape;

}
```

В данном методе мы задаем фигуру ИИ рандомно, а фигуру для игрока отлавливаем с нажатия на кнопку. После того как фигуры определены мы делаем проверку на победителя и выбираем соответствующее сообщение для игрока.

```
// Run game action
@Override
public void actionPerformed(ActionEvent e)
{

    // Generate computer/player step
    GameShape computerShape = generateShape();
    GameShape playerShape = new GameShape();


    // Read the button is pressed and define figure
    switch (e.getActionCommand())
    {

        case "rock":
            playerShape = new Rock();
            break;

        case "paper":
            playerShape = new Paper();
            break;

        case "scissors":
            playerShape = new Scissors();
            break;

    }

    // Check if we have winner
    int gameResult = checkWinner(playerShape, computerShape);

    // Getting message for player
    String message = "Player shape: " + playerShape + ". Computer shape: " +
computerShape + ". ";
    switch (gameResult)
    {

        case -1:
            message += "Computer has won!";
            break;
```

```
        case 0:
            message += "It's a tie!";
            break;

        case 1:
            message += "Player has won!";

    }

    // Print message
    JOptionPane.showMessageDialog(null  message);

}
```

4. Модификация программы Draw

Создаем класс эллипса чтобы для отрисовки

```
public class Ellipse extends DrawShape
{

    public Ellipse(){}

    public Ellipse(Point startPoint  Point endPoint)
    {
        super(startPoint  endPoint);
    }

    @Override
    public Shape getShape(Point startPoint  Point endPoint)
    {

        return new Ellipse2D.Double(Math.min(startPoint.getX()
                                    endPoint.getX())
                                    Math.min(startPoint.getY()
                                    endPoint.getY())
                                    Math.abs(startPoint.getX() -
                                    endPoint.getX())
                                    Math.abs(startPoint.getY() -
                                    endPoint.getY()));

    }

}
```

Добавляем кнопку для эллипса

```
// **Ellipse button
BigTextButton ellipse = new BigTextButton("Ellipse");

// Notify surface that it draws ellipses now
ellipse.addActionListener(e -> { surface.setShapeType(DrawShape.SHAPE_ELLIPSE);
});

// Add button to the panel
buttonPanel.add(ellipse);
```

Метод очистки работает достаточно просто, нам нужно просто очистить наш массив фигур и перерисовать спейс

```
// Clear surface
public void clear()
{
    shapes.clear();
    repaint();
}
```

Добавляем кнопку для очистки спейса и говорим что при нажатии на эту кнопку нужно переопределить метод отлова action и чтобы он выполнил очистку массива фигур.

```
// **Clear button
BigTextButton clear = new BigTextButton("Clear");

// Notify surface that it must be cleared
clear.addActionListener(new ActionListener()
{
    @Override
    public void actionPerformed(ActionEvent e) { surface.clear();}
});

// Add button to the panel
buttonPanel.add(clear);
```

Выводы: более детально изучено механизмы наследования и полиморфизма в Java.