**Белорусский национальный технический университет**

**Кафедра «Программное обеспечение информационных систем и технологий»**

**Дисциплина «Программирование на языке Java»**

**Лабораторная работа №5**

Выполнили: студенты гр. 10702122 Тимощенко Н.В   
 Кудрявец Д.Е

Проверила: пр.- ст. Савчик А.

Минск 2024

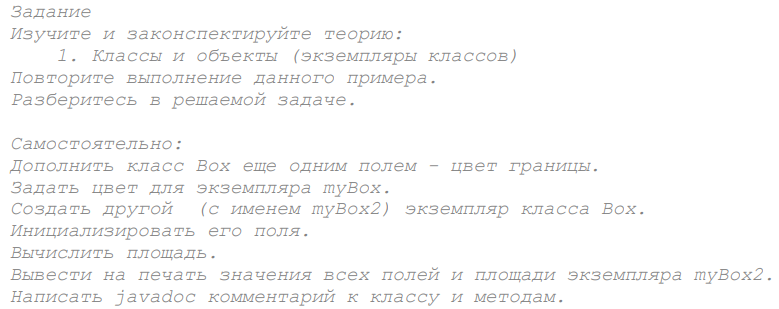
**Лабораторная работа 5**

**BoxDemo**

**Цель работы:**

Изучить классы и объекты(экземпляры классов).

**Задание:**

****

**Код программы:**

*/\*\*  
 \* @author Tsimoshchenko M.V, Kudryavets D.E  
 \* @version 1.0  
 \*/*public class BoxDemo {  
 public static void main(String[] args){  
 Box myBox=new Box();  
 double area;  
  
 myBox.length=15;  
 myBox.width=10;  
 area=myBox.length \* myBox.width;  
 System.*out*.println("Square = "+area);  
  
 Box myBox2=new Box();  
 myBox2.length=20;  
 myBox2.width=15;  
 myBox2.borderColor="Red";  
 area=myBox2.length \* myBox2.width;  
 myBox2.Print();  
 System.*out*.println("Square = "+area);  
 }  
}  
  
class Box {  
 double width;  
 double length;  
 String borderColor;  
  
 public void Print(){  
 System.*out*.println("Width = "+width);  
 System.*out*.println("Length = "+length);  
 System.*out*.println("Border's color = "+borderColor);  
 }  
}

**Скриншоты выполнения программы:**

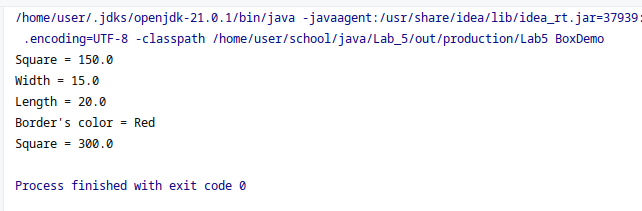


Рис. 1 – выполнение программы BoxDemo

**Контрольные вопросы:**

*1) Как создаются экземпляры класса?*

Для начала необходимо определить класс, который необходимо создать,к примеру:

public class Dog {

String name;

// Конструктор

public Dog(String name) {

this.name = name;

}

public void bark() {

System.out.println(name + " says woof!");

}

}

И уже далее создается экземпляр класса. Для этого используется

оператор new за которым следует имя класса и его конструктор:

public class Main {

public static void main(String[] args) {

Dog myDog = new Dog("Buddy"); // Создание экземпляра Dog

myDog.bark(); // Вызов метода bark

}

}

*2) Как изменить поля экземпляра класса?*

В Java для изменения полей экземпляра класса вы можете использовать методы или напрямую обращаться к полям, если они имеют модификатор доступа, позволяющий это. Вот несколько способов, как это можно сделать:

1)Прямой доступ к полям. Если поля объявлены как public, вы сможете изменить их напрямую:

public class Dog {

public String name; // Поле public

public Dog(String name) {

this.name = name;

}

}

public class Main {

public static void main(String[] args) {

Dog myDog = new Dog("Buddy");

myDog.name = "Max"; // Изменение поля напрямую

System.out.println(myDog.name); // Вывод: Max

}

}

2) Использование методов(get и set). Рекомендуется использовать методы для изменения полей, особенно если они имеют модификатор доступа private. Это обеспечит инкапсуляцию данных:

public class Dog {

private String name; // Поле private

public Dog(String name) {

this.name = name;

}

// Метод для изменения имени

public void setName(String name) {

this.name = name;

}

// Метод для получения имени

public String getName() {

return name;

}

}

public class Main {

public static void main(String[] args) {

Dog myDog = new Dog("Buddy");

myDog.setName("Max"); // Изменение поля через метод

System.out.println(myDog.getName()); // Вывод: Max

}

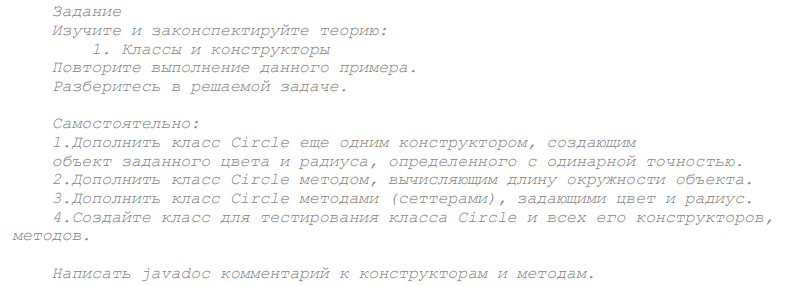
}

**TestCircle**

**Цель работы:**

Изучить классы и конструкторы.

**Задание:**

****

**Код программы:**

*/\*\*  
 \* @author Tsimoshchenko M.V, Kudryavets D.E  
 \* @version 1.0  
 \*/*public class CircleTest {  
 public static void main(String[] args) {  
 Circle circle1=new Circle();  
 System.*out*.println("Circle 1 - Radius: "+circle1.getRadius()+", Color: "+circle1.getColor());  
 System.*out*.println("Area: "+circle1.getArea());  
 System.*out*.println("Length: "+circle1.getLength());  
  
 Circle circle2=new Circle(2.0);  
 System.*out*.println("Circle 2 - Radius: "+circle2.getRadius()+", Color: "+circle2.getColor());  
 System.*out*.println("Area: "+circle2.getArea());  
 System.*out*.println("Length: "+circle2.getLength());  
  
 Circle circle3=new Circle(2.5, "Green");  
 System.*out*.println("Circle 3 - Radius: "+circle3.getRadius()+", Color: "+circle3.getColor());  
 System.*out*.println("Area: "+circle3.getArea());  
 System.*out*.println("Length: "+circle3.getLength());  
  
 Circle circle4=new Circle(2.5f, "Yellow");  
 System.*out*.println("Circle 4 - Radius: "+circle4.getRadius()+", Color: "+circle4.getColor());  
 System.*out*.println("Area: "+circle4.getArea());  
 System.*out*.println("Length: "+circle4.getLength());  
  
 circle1.SetRadius(2.5);  
 circle1.SetColor("Red");  
 System.*out*.println("Circle 1 after setting new values - Radius: "+circle1.GetRadius()+", Color: "+circle1.GetColor());  
 System.*out*.println("Area: "+circle1.getArea());  
 System.*out*.println("Length: "+circle1.getLength());  
 }  
}  
  
class Circle {  
 private double radius;  
 private String color;  
  
 public Circle() {  
 radius=3.0;  
 color="Red";  
 }  
  
 public Circle(double r) {  
 radius=r;  
 color="Red";  
 }  
  
 public Circle(double r, String c) {  
 radius=r;  
 color=c;  
 }  
  
 public Circle(float r, String c) {  
 radius=r;  
 color=c;  
 }  
  
 *// Public methods* public double getRadius() {  
 return radius;  
 }  
  
 public String getColor() {  
 return color;  
 }  
  
 public double getArea() {  
 return radius \* radius \* Math.*PI*;  
 }  
  
 public double getLength() {  
 return Math.*PI* \* radius \* 2;  
 }  
  
 *//setters* public void SetRadius(double radius) {  
 this.radius=radius;  
 }  
  
 public void SetColor(String color) {  
 this.color=color;  
 }  
  
 *//getters* public double GetRadius() {  
 return this.radius;  
 }  
  
 public String GetColor() {  
 return this.color;  
 }  
}

**Скриншоты выполнения программы:**

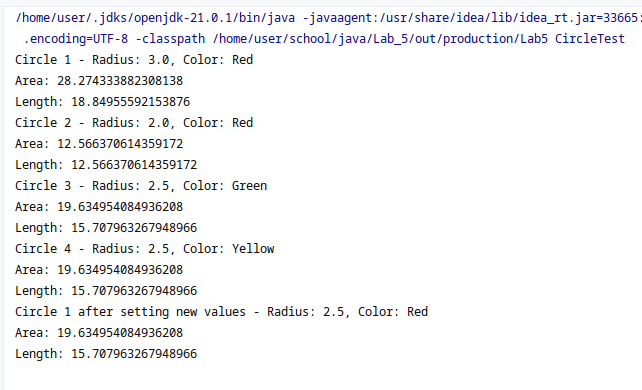


Рис. 2 – выполнение программы TestCircle

**Контрольные вопросы:**

*1)Что такое конструктор по умолчанию?*

Конструктор по умолчанию — это специальный тип конструктора в Java, который автоматически создается компилятором, если вы не определяете ни одного конструктора в классе. Он инициализирует объект значениями по умолчанию:

* Для примитивных типов (например, int, double) — нулевыми значениями.
* Для объектов — значением null.

Пример конструктора по умолчанию представлен ниже:

public class Dog {

String name; // Поле класса

// Конструктор по умолчанию создается автоматически

}

public class Main {

public static void main(String[] args) {

Dog myDog = new Dog(); // Используем конструктор по умолчанию

System.out.println(myDog.name); // Вывод: null

}

}

*2)Какие модификаторы доступа рекомендуются использовать для полей, методов?*

В Java существуют четыре основных модификатора доступа:

1. Public

* Доступ: Члены класса доступны из любого другого класса в любом пакете.
* Использование: Для элементов, которые должны быть доступны повсеместно.

1. Private

* Доступ: Члены класса доступны только внутри самого класса.
* Использование: Для сокрытия данных и обеспечения инкапсуляции.

1. Protected

* Доступ: Члены класса доступны внутри своего класса, в подклассах и в классах того же пакета.
* Использование: Для элементов, которые должны быть доступны в наследниках и классах одного пакета.

1. Default (пакетный доступ)

* Доступ: Члены класса доступны только внутри того же пакета, если модификатор не указан.
* Использование: Для элементов, доступных только внутри одного пакета.

С точки зрения объектно-ориентированного программирования (ООП)рекомендуется использовать модификатор доступа private, так как Сокрытие полей с помощью модификатора private позволяет защитить данные от прямого доступа извне, что способствует инкапсуляции. Это предотвращает случайные изменения состояния объекта и позволяет контролировать, как данные используются и изменяются.

*3) Что такое инкапсуляция?*

Инкапсуляция — это один из ключевых принципов объектно-ориентированного программирования (ООП), который заключается в сокрытии внутреннего состояния объекта и ограничения доступа к его данным. В Java инкапсуляция достигается с помощью модификаторов доступа и методов (геттеров и сеттеров). Основными аспектами инкапсуляции являются:

1) Сокрытие данных;

2) Использование методов доступа (get, set);

3) Управление доступом.

**Глоссарий:**

1. Инкапсуляция – это принцип объектно-ориентированного программирования, который заключается в сокрытии внутреннего состояния объекта и ограничения доступа к его данным. Это достигается с помощью сокрытия данных, использования методов: геттеров и сеттеров.

2. Конструктор по умолчанию – это специальный тип конструктора в Java, который автоматически создается компилятором, если в классе не определено ни одного конструктора. Он инициализирует объект значениями по умолчанию: для примитивных типов (например, int, double) — нулевыми значениями, а для объектов — значением null.

3. Класс – это основная конструкция, используемая для создания объектов. Он представляет собой шаблон или чертеж, который описывает свойства (поля) и поведение (методы) объектов. Классы являются основой объектно-ориентированного программирования (ООП) в Java.

4. Метод – блок программного кода, относящегося к описанию поведения объекта или его класса.

5. Поле экземпляра класса – это переменная, которая объявлена внутри класса и представляет собой состояние конкретного объекта (экземпляра) этого класса. Поля экземпляра используются для хранения данных, которые принадлежат каждому отдельному объекту.

6. Модификаторы доступа – это ключевые слова, которые определяют уровень доступа к классам, полям и методам. Они помогают контролировать, откуда можно получить доступ к различным элементам кода, что способствует инкапсуляции и безопасности данных. Существуют четыре основных модификатора доступа: public, private, protected, default.

7. Экземпляр класса – это конкретный объект, созданный на основе определения данного класса. Каждый экземпляр класса имеет свои собственные значения полей и может выполнять методы, описанные в классе. Экземпляры позволяют использовать один и тот же шаблон (класс) для создания множества объектов с различными состояниями и поведением.