****Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Кафедра автоматизації проектування енергетичних процесів і систем

ЗВІТ   
про виконання лабораторної роботи №5  
з дисципліни «Технології конструювання програмного забезпечення»

**«Работа с классами и их документирование»**

Виконав: студент групи TI-92

Черноусов Денис Ігорович

Перевірив: доцент, к.ф.-м.н.  
Тарнавський Ю.А.

Київ - 2020

# Ход выполнения работы

1. Загрузите IntelliJ IDEA и настройте ее, чтобы подсказка Javadoc возникала при перемещении мыши над именем класса (метода).
2. Установите плагин simpleUMLCE.
3. Создайте проект Maven.
4. Создайте класс в соответствии с вариантом индивидуального задания. Класс следует унаследовать от интерфейса. Реализуйте для него хотя бы 2 конструктора и клонирование, переопределите методы equals(), hashCode(), toString(). Обеспечьте инкапсуляцию данных.
5. Создайте unit-тесты для тестирования методов класса и выполните их тестирование.
6. Если обнаружены методы, не прошедшие тест, внесите необходимые изменения в код и повторите тест.
7. Сформируйте диаграмму класса.
8. Вставьте необходимые комментарии и сгенерируйте справку Javadoc.

**16 варіант**

5. Построить программу для работы с классом для хранения данных о сообщении на форуме (автор, тема, текст, время, дата создания и редактирование). Программа должна обеспечивать простейшие функции: ввод значений, редактирование, вывод значений.

13. Создать класс Квадрат. Поля - сторона. Функции-члены вычисляют площадь, периметр, устанавливает поля и возвращают значения. Функции-члены установки полей класса должны проверять корректность задаваемых параметров.

25. Создать базовый класс – «полином» (массив коэффициентов), производный класс «рациональное выражение», включающий полином в числителе, полином в знаменателе.

31. Создать класс машина, имеющий марку, число цилиндров, мощность. Создать производный класс - грузовик, имеющий грузоподъемность.

Завдання 5

Клас Message

package task5;  
  
import java.util.Objects;  
  
*/\*\*  
 \* Class for storing data of messages on a forum  
 \*  
 \* <b>Author</b>, <b>Theme</b>, <b>Text</b>,  
 \* <b>Time</b>, <b>Redaction</b>  
 \*  
 \** ***@author*** *Denis  
 \** ***@version*** *1.0  
 \*/*public class Message implements Forum, Cloneable {  
  
 */\*\*  
 \* Author of a message  
 \*/* private String Author;  
 */\*\*  
 \* Theme of a message  
 \*/* private String Theme;  
 */\*\*  
 \* Text of a message  
 \*/* private String Text;  
 */\*\*  
 \* Creation time of a message  
 \*/* private String Time;  
 */\*\*  
 \* Redaction of a message  
 \*/* private int Redaction;  
  
 */\*\*  
 \* Constructor with values  
 \*  
 \** ***@param*** *author  
 \** ***@param*** *theme  
 \** ***@param*** *text  
 \** ***@param*** *time  
 \** ***@param*** *redaction  
 \** ***@see*** *Message#Message(String, String, String, String, int)  
 \*/* public Message(String author, String theme, String text, String time, int redaction) {  
 Author = author;  
 Theme = theme;  
 Text = text;  
 Time = time;  
 Redaction = redaction;  
 }  
  
 */\*\*  
 \* Constructor with only one value  
 \*  
 \** ***@param*** *text  
 \** ***@see*** *Message#Message(String) String of text  
 \*/* public Message(String text) {  
 Text = text;  
 }  
  
 */\*\*  
 \* Returns data of an another object to this object  
 \*/* @Override  
 protected Message clone() throws CloneNotSupportedException {  
  
 return (Message) super.clone();  
  
 }  
  
 */\*\*  
 \* Compares two objects.  
 \*  
 \** ***@param*** *o  
 \** ***@return*** *True , if they are equal . Otherwise - false .  
 \*/* @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Message)) return false;  
 Message Message = (Message) o;  
 return Objects.*equals*(getAuthor(), Message.getAuthor()) &&  
 Objects.*equals*(getTheme(), Message.getTheme()) &&  
 Objects.*equals*(getText(), Message.getText()) &&  
 Objects.*equals*(getTime(), Message.getTime()) &&  
 Objects.*equals*(getRedaction(), Message.getRedaction());  
 }  
  
 */\*\*  
 \* Gets an unique integer from integer.  
 \*  
 \** ***@return*** *Hash integer of a message.  
 \*/* @Override  
 public int hashCode() {  
 return Objects.*hash*(getAuthor(), getTheme(), getText(), getTime(), getRedaction());  
 }  
  
 */\*\*  
 \* Shows all information about a message.  
 \*  
 \** ***@return*** *All information about a message.  
 \*/* @Override  
 public String toString() {  
 return "Forum{" +  
 "Author='" + Author + '\'' +  
 ", Theme='" + Theme + '\'' +  
 ", Text=" + Text +  
 ", Time='" + Time + '\'' +  
 ", Redaction='" + Redaction + '\'' +  
 '}';  
 }  
  
 */\*\*  
 \* Method for getting a field {****@link*** *Message#Author}  
 \*  
 \** ***@return*** *A name of an authors .  
 \*/* public String getAuthor() {  
 return Author;  
 }  
  
 public void setAuthor(String author) {  
 Author = author;  
 Redaction++;  
 }  
  
 */\*\*  
 \* Method for getting a field {****@link*** *Message#Theme}  
 \*  
 \** ***@return*** *A title of an theme .  
 \*/* public String getTheme() {  
 return Theme;  
 }  
  
 public void setTheme(String theme) {  
 Theme = theme;  
 Redaction++;  
 }  
  
 */\*\*  
 \* Method for getting a field {****@link*** *Message#Text}  
 \*  
 \** ***@return*** *Text of a message .  
 \*/* public String getText() {  
 return Text;  
 }  
  
 public void setText(String text) {  
 Text = text;  
 Redaction++;  
 }  
  
 */\*\*  
 \* Method for getting a field {****@link*** *Message#Time}  
 \*  
 \** ***@return*** *Time of bringing out .  
 \*/* public String getTime() {  
 return Time;  
 }  
  
 public void setTime(String time) {  
 Time = time;  
 Redaction++;  
 }  
  
 */\*\*  
 \* Method for getting a field {****@link*** *Message#Redaction}  
 \*  
 \** ***@return*** *Redaction of a massage.  
 \*/* public int getRedaction() {  
 return Redaction;  
 }  
  
 public void setRedaction(int redaction) {  
 Redaction = redaction;  
 }  
  
}

Інтерфейс Forum

package task5;  
  
public interface Forum {  
  
 */\*\*  
 \* Outputs data  
 \*/* String toString();  
  
 */\*\*  
 \* Inputs data  
 \*/* void setAuthor(String author);  
  
 void setTheme(String theme);  
  
 void setText(String text);  
  
 void setTime(String time);  
  
 void setRedaction(int redaction);  
  
}

Клас Main

package task5;  
  
import java.util.Calendar;  
  
  
public class Main {  
 public static void main(String[] args) throws CloneNotSupportedException {  
  
 Forum m1 = new Message("Simple text");  
  
 m1.setAuthor("Vasya");  
  
 System.*out*.println(m1);  
  
 Calendar cal1 = Calendar.*getInstance*();  
  
 Forum m2 = new Message("Vasya", "Interface", "Another text", cal1.getTime().toString(), 1);  
  
 System.*out*.println(m2);  
  
 m1 = ((Message) m2).clone();  
  
 m1.setAuthor("Denis");  
  
 System.*out*.println(m1);  
  
 */\*\*  
 \* java.lang.Objects are used to compare these instances  
 \*/* if (!m2.equals(m1)) System.*out*.println("java.lang.Objects are used to compare these instances");  
  
 }  
}

Вивід

Forum{Author='Vasya', Theme='null', Text=Simple text, Time='null', Redaction='1'}

Forum{Author='Vasya', Theme='Interface', Text=Another text, Time='Thu Nov 05 12:18:07 EET 2020', Redaction='1'}

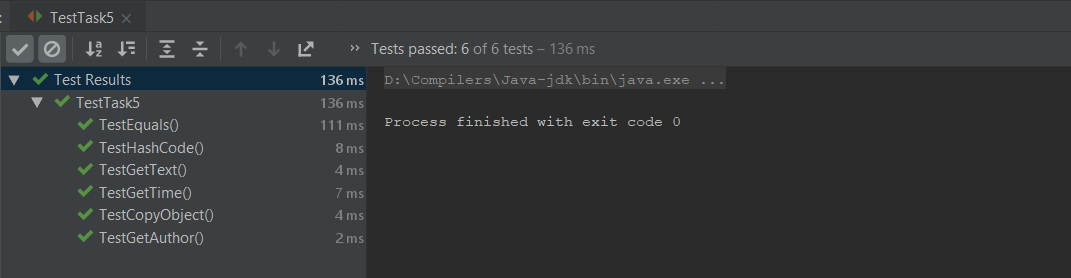
Forum{Author='Denis', Theme='Interface', Text=Another text, Time='Thu Nov 05 12:18:07 EET 2020', Redaction='2'}

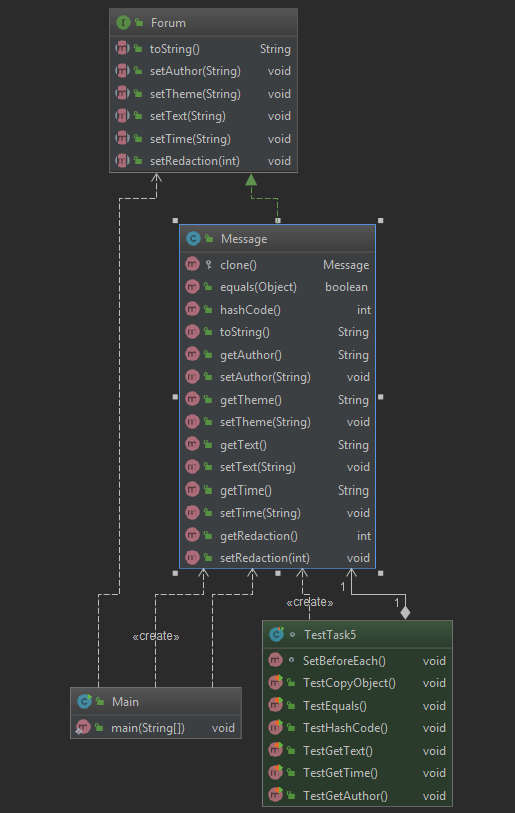
java.lang.Objects are used to compare these instances

Тести до завдання

package task5;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import java.util.Calendar;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class TestTask5 {  
  
 Message m1;  
 Message m2;  
 Calendar cal1 ;  
  
 @BeforeEach  
 void SetBeforeEach() {  
 m1 = new Message("Some text");  
  
 cal1 = Calendar.*getInstance*();  
  
 m2 = new Message("Vasya", "Interface", null,  
 cal1.getTime().toString(), 1);  
 }  
  
 @Test  
 void TestCopyObject() throws CloneNotSupportedException{  
  
 m2 = m1.clone();  
  
 *assertEquals*(m1, m2);  
  
 }  
  
 @Test  
 void TestEquals() {  
  
 *assertEquals*(false, m1.equals(m2));  
 }  
  
 @Test  
 void TestHashCode() {  
 int hash = m1.hashCode() ;  
 *assertEquals*(hash, m1.hashCode());  
 }  
  
 @Test  
 void TestGetText() {  
  
 *assertEquals*("Some text", m1.getText());  
  
 }  
  
 @Test  
 void TestGetTime() {  
  
 *assertEquals*(cal1.getTime().toString(), m2.getTime());  
  
 }  
  
 @Test  
 void TestGetAuthor() {  
  
 *assertEquals*("Vasya", m2.getAuthor());  
  
 }  
  
}

Вивід



Діаграма

Завдання 13

Клас Quadrant

package task13;  
  
import java.util.Objects;  
  
*/\*\*  
 \* Class for calculating data of Quadrant  
 \*  
 \* <b>side</b>, <b>perimeter</b>, <b>square</b>,  
 \*  
 \** ***@author*** *Denis  
 \** ***@version*** *1.0  
 \*/*public class Quadrant implements Measurable {  
 */\*\*  
 \* one side of Quadrant  
 \*/* private float side;  
 */\*\*  
 \* perimeter of Quadrant  
 \*/* private float perimeter;  
 */\*\*  
 \* square of Quadrant  
 \*/* private float square;  
  
 */\*\*  
 \* Checks bounds and returns value of Parameter .  
 \*  
 \** ***@param*** *Parameter should be more than 0 to be returned  
 \** ***@return*** *0, if Parameter less or equal 0.  
 \*/* private float checkBoundsAndReturn(float Parameter) {  
 if (Parameter <= 0) {  
 System.*out*.println(Parameter + " is out of range! Set parameter > 0 !");  
 return 0;  
 } else return Parameter;  
 }  
  
 */\*\*  
 \* Calculates perimeter and square by side of Quadrant  
 \*/* private void findPerimeterAndSquare() {  
 perimeter = 4 \* side;  
 square = side \* side;  
 }  
  
 */\*\*  
 \* Constructor with value side  
 \*/* public Quadrant(float side) {  
 this.side = checkBoundsAndReturn(side);  
 findPerimeterAndSquare();  
 }  
  
 */\*\*  
 \* Constructor without values  
 \*/* public Quadrant() {  
 System.*out*.println("Please, set a side of Quadrant by setSide().");  
 }  
  
 */\*\*  
 \* Outputs all information of object  
 \*  
 \** ***@return*** *String of values  
 \*/* @Override  
 public String toString() {  
 return "Quadrant{" +  
 "side=" + getSide() +  
 ", perimeter=" + getPerimeter() +  
 ", square=" + getSquare() +  
 '}';  
 }  
  
 */\*\*  
 \*  
 \*  
 \** ***@return*** *Quadrant with values  
 \*/* public Quadrant clone() {  
 return new Quadrant(this.side);  
 }  
  
 */\*\*  
 \* Checks if Objects are equal.  
 \*  
 \** ***@param*** *o Object  
 \** ***@return*** *true if they are equal , false if not .  
 \*/* @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Quadrant)) return false;  
 Quadrant quadrant = (Quadrant) o;  
 return Float.*compare*(quadrant.getSide(), getSide()) == 0 &&  
 Float.*compare*(quadrant.getPerimeter(), getPerimeter()) == 0 &&  
 Float.*compare*(quadrant.getSquare(), getSquare()) == 0;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(getSide(), getPerimeter(), getSquare());  
 }  
  
 public float getSide() {  
 return side;  
 }  
  
 public float getSquare() {  
 return square;  
 }  
  
 public float getPerimeter() {  
 return perimeter;  
 }  
  
  
 public void setSide(float side) {  
 this.side = checkBoundsAndReturn(side);  
 findPerimeterAndSquare();  
 }  
  
 public void setPerimeter(float perimeter) {  
 this.perimeter = checkBoundsAndReturn(perimeter);  
 }  
  
 public void setSquare(float square) {  
 this.square = checkBoundsAndReturn(square);  
 }  
  
}

Інтерфейс Measurable

package task13;  
  
public interface Measurable {  
  
 void setSide(float side);  
  
 void setSquare(float Square);  
  
 void setPerimeter(float Perimeter);  
  
 Object clone();  
  
  
}

Клас Main

package task13;  
  
public class Main {  
 public static void main(String[] args) {  
  
 Measurable o1 = new Quadrant();  
  
 System.*out*.println(o1);  
  
 o1.setSide(-1);  
  
 System.*out*.println(o1);  
  
 o1.setSide(10);  
  
 System.*out*.println(o1);  
  
 Measurable o2 = (Quadrant) o1.clone();  
  
 System.*out*.println(o2);  
  
  
 }  
}

Вивід

Please, set a side of Quadrant by setSide().

Quadrant{side=0.0, perimeter=0.0, square=0.0}

-1.0 is out of range! Set parameter > 0 !

Quadrant{side=0.0, perimeter=0.0, square=0.0}

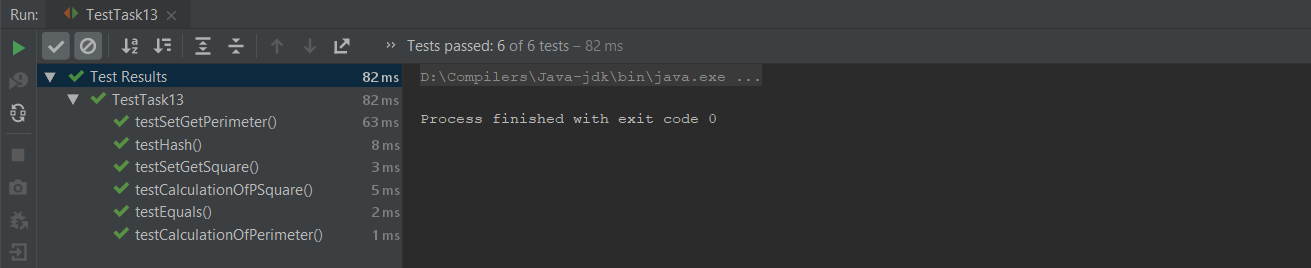
Quadrant{side=10.0, perimeter=40.0, square=100.0}

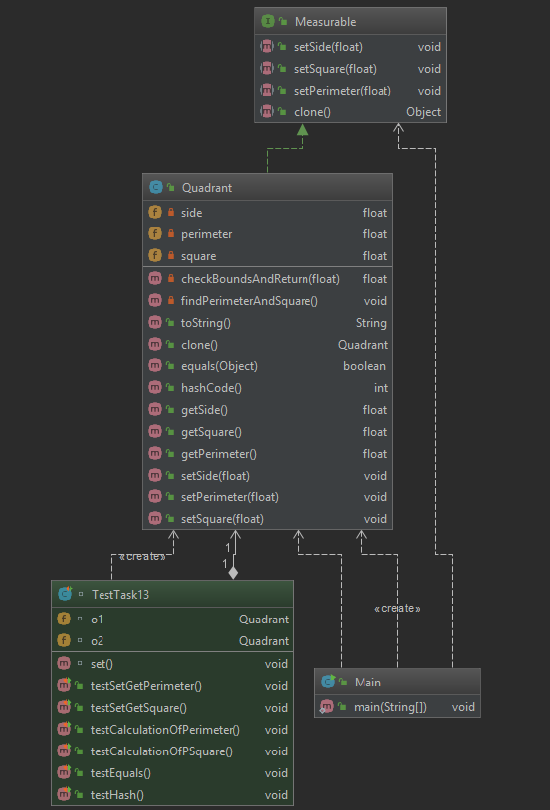
Quadrant{side=10.0, perimeter=40.0, square=100.0}

Тест до завдання

package task13;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class TestTask13 {  
  
 Quadrant o1;  
 Quadrant o2;  
  
 @BeforeEach  
 void set() {  
 o1 = new Quadrant(1);  
 o2 = new Quadrant(10);  
 }  
  
 @Test  
 void testSetGetPerimeter() {  
 o1.setPerimeter(10);  
 *assertEquals*(10, o1.getPerimeter());  
 }  
  
 @Test  
 void testSetGetSquare() {  
 o1.setSquare(10);  
 *assertEquals*(10, o1.getSquare());  
 }  
  
 @Test  
 void testCalculationOfPerimeter() {  
 *assertEquals*(40, o2.getPerimeter());  
 }  
  
 @Test  
 void testCalculationOfPSquare() {  
 *assertEquals*(100, o2.getSquare());  
 }  
  
 @Test  
 void testEquals() {  
 *assertEquals*(false, o2.equals(o1));  
 }  
  
 @Test  
 void testHash() {  
 *assertEquals*(false, o2.hashCode() == o1.hashCode());  
 }  
  
}

Вивід



Діаграма

Завдання 25

Клас Polynomial

package task25;  
  
*/\*\*  
 \* Class for storing data of Polynomial  
 \*  
 \* <b>values</b>, <b>length</b>  
 \*  
 \** ***@author*** *Denis  
 \** ***@version*** *1.0  
 \*/*public class Polynomial {  
  
 */\*\* Array of coefficients \*/* private double values[];  
  
 */\*\* length of Array of coefficients \*/* private int length;  
  
 */\*\*  
 \* Constructor for polynomial  
 \*  
 \** ***@param*** *values  
 \*/* public Polynomial(double[] values) {  
  
 this.values = values;  
 this.length = values.length;  
 }  
  
 */\*\*Returns length of Array of coefficients  
 \*  
 \** ***@return*** *length  
 \*/* public int getLength() {  
 return length;  
 }  
  
 */\*\*Returns Array of coefficients  
 \*  
 \** ***@return*** *array of values  
 \*/* public double[] getValues() {  
 return values;  
 }  
  
 */\*\*  
 \* Sets new coefficients in Polynomial and sets a new length.  
 \*  
 \** ***@param*** *values  
 \*/* public void setValues(double[] values) {  
 this.length = values.length;  
 this.values = values;  
 }  
  
  
}

Клас Rational

package task25;  
  
import java.util.Arrays;  
import java.util.Objects;  
  
*/\*\*  
 \* Class for storing data of Rational.Subclass of Polynomial.  
 \*  
 \* <b>divider</b>  
 \*  
 \** ***@author*** *Denis  
 \** ***@version*** *1.0  
 \*/*public class Rational extends Polynomial implements Countable {  
  
 */\*\* Polynomial divider \*/* private Polynomial divider;  
  
 */\*\*  
 \* Constructor for numerator and divider  
 \*  
 \** ***@param*** *numerator  
 \** ***@param*** *dividerValues  
 \*/* Rational(double[] numerator, double[] dividerValues) {  
 super(numerator);  
 this.divider = new Polynomial(dividerValues);  
 checkDividerValues();  
 }  
  
 */\*\*  
 \* Constructor for numerator  
 \*  
 \** ***@param*** *numerator  
 \*/* Rational(double[] numerator) {  
 super(numerator);  
 this.divider = new Polynomial(new double[]{1});  
 }  
  
  
 */\*\*  
 \* Clones Rational  
 \*  
 \** ***@return*** *\*/* @Override  
 public Rational clone() {  
 return new Rational(getValues(), getDividerValues());  
 }  
  
 */\*\*  
 \* Checks if a divider is a zero array and sets a new value if it is.  
 \*/* private void checkDividerValues() {  
  
 if (Arrays.*equals*(new double[4], getDividerValues())) {  
 System.*out*.println("It can't be a zero divider! Setting a new value.");  
 setDividerValues(new double[]{1});  
 }  
 }  
  
 */\*\*  
 \* prints Polynomial  
 \*  
 \** ***@param*** *p  
 \*/* private void printPolynomial(Polynomial p) {  
  
 System.*out*.printf("%.0f", p.getValues()[0]);  
  
 for (int i = 1; i < p.getLength(); i++) {  
  
 if (i == 1 && p.getValues()[i] == 1) System.*out*.printf(" + x");  
  
 else if (i == 1 && p.getValues()[i] != 0) System.out.printf(" + %.0fx", p.getValues()[i]);  
  
 else if (p.getValues()[i] != 0 && p.getValues()[i] != 1)  
 System.out.printf(" + %.0fx^%d", p.getValues()[i], i);  
  
 else if (p.getValues()[i] == 1) System.out.printf(" + x^%d", i);  
  
 }  
  
 }  
  
 */\*\*  
 \* Outputs rational polynomial (numerator and divider).  
 \*/* public void output() {  
  
 System.out.print("( ");  
  
 printPolynomial(this);  
  
 System.out.print(" ) / ( ");  
  
 printPolynomial(divider);  
  
 System.out.println(" )");  
  
 }  
  
 */\*\* sets a new divider coefficients and cheks it.  
 \*  
 \** ***@param*** *values  
 \*/* public void setDividerValues(double[] values) {  
 divider.setValues(values);  
 checkDividerValues();  
 }  
  
 */\*\*  
 \* gets dividers coefficients.  
 \** ***@return*** *\*/* public double[] getDividerValues() {  
 return divider.getValues();  
 }  
  
 */\*\*  
 \* gets dividers length.  
 \** ***@return*** *\*/* public int getDividerLength() {  
 return divider.getLength();  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *String  
 \*/* @Override  
 public String toString() {  
 return "Rational{" +  
 " numerator values = " + Arrays.toString(getValues()) +  
 ", numerator length = " + getLength() +  
 ", divider values = " + Arrays.toString(getDividerValues()) +  
 ", divider length = " + getDividerLength() +  
 '}';  
 }  
  
 */\*\*  
 \* Compare two rationals  
 \*  
 \** ***@param*** *o  
 \** ***@return*** *\*/* @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Rational)) return false;  
 Rational rational = (Rational) o;  
 return getLength() == rational.getLength() &&  
 getDividerLength() == rational.getDividerLength() &&  
 Arrays.equals(getValues(), rational.getValues()) &&  
 Arrays.equals(getDividerValues(), rational.getDividerValues());  
 }  
  
 */\*\*  
 \* get hash code from rational.  
 \** ***@return*** *\*/* @Override  
 public int hashCode() {  
 return Objects.hash(getLength(), getValues(), getDividerValues(), getDividerLength());  
 }  
}

Інтерфейс Countable

package task25;  
  
public interface Countable {  
  
 void setDividerValues(double[] Values);  
  
 void setValues(double[] Values);  
  
 void output();  
  
 int getDividerLength();  
  
 int getLength();  
}

Клас Main

package task25;  
  
public class Main {  
 public static void main(String[] args) {  
  
 double arr[] = {1, 2, 0, 5};  
 Countable p = new Rational(arr);  
  
 System.*out*.println(p);  
  
 p.setValues(new double[]{0, 0, 0, 1, 0, 1, 2, 0, 1});  
  
 p.setDividerValues(new double[4]);  
  
 System.*out*.println(p);  
  
 p.output();  
  
 Countable p2 = new Rational(new double[]{1, 2 , 0 , 1} , new double[]{0, 1 , 3 , 2 } );  
  
 System.*out*.println(p2);  
  
 p = ((Rational) p2).clone();  
  
 if (p2.equals(p)) p.output();  
  
  
 }  
}

Вивід

Sodom is new in the Gallery

Dali already is in the Gallery

(type "out" to quit) Check our Gallery :

Liza Walli Dali Sodom Scream Elon

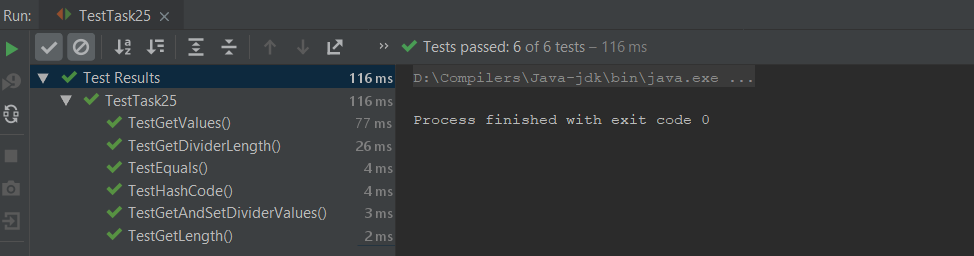
Liza

Out

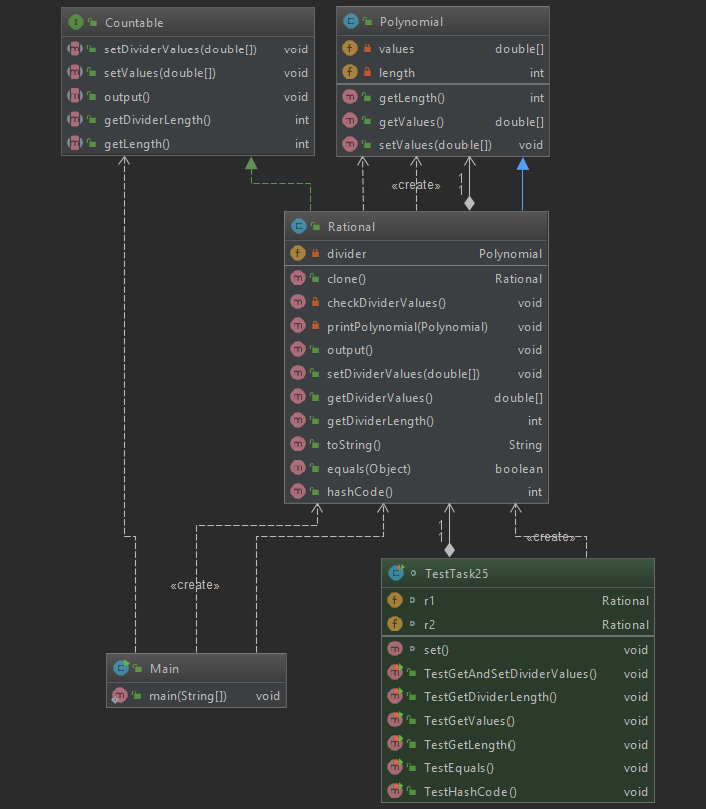
Тести до завдання

package task25;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class TestTask25 {  
  
 Rational r1;  
 Rational r2 ;  
  
 @BeforeEach  
 void set(){  
 r1 = new Rational(new double[]{1,2});  
 r1.setDividerValues(new double[]{1, 2, 3});  
 r2 = r1.clone();  
 }  
  
 @Test  
 void TestGetAndSetDividerValues() {  
  
 *assertArrayEquals*(new double[]{1, 2, 3}, r1.getDividerValues());  
 }  
  
 @Test  
 void TestGetDividerLength(){  
 *assertEquals*(3,r1.getDividerLength());  
 }  
  
 @Test  
 void TestGetValues(){  
 *assertArrayEquals*(new double[]{1,2} , r1.getValues());  
 }  
  
 @Test  
 void TestGetLength(){  
 *assertEquals*(2 , r1.getLength());  
 }  
  
 @Test  
 void TestEquals(){  
 *assertEquals*(r2 , r1);  
 }  
  
 @Test  
 void TestHashCode(){  
 *assertEquals*(r2.hashCode() , r1.hashCode());  
 }  
  
}

Вивід



Діаграма



Завдання 31

Клас Car

package task31;  
  
*/\*\*  
 \* Class for storing data of Car  
 \*  
 \* <b>power</b>, <b>brand</b> <b>Cylinder</b>  
 \*  
 \** ***@author*** *Denis  
 \** ***@version*** *1.0  
 \*/*public class Car {  
 */\*\*  
 \* power of car  
 \*/* private int power;  
 */\*\*  
 \* brand name of car  
 \*/* private String brand;  
 */\*\*  
 \* number of cylinders of car  
 \*/* private int cylinder;  
  
 */\*\*  
 \* Constructor  
 \*  
 \** ***@param*** *power  
 \** ***@param*** *brand  
 \** ***@param*** *cylinder  
 \*/* public Car(int power, String brand, int cylinder) {  
 setPower(power);  
 setBrand(brand);  
 setCylinder(cylinder);  
 }  
  
 */\*\*  
 \** ***@return*** *power of car  
 \*/* public int getPower() {  
 return power;  
 }  
  
 */\*\*  
 \* Sets a new power of car if it is more than 0.  
 \*  
 \** ***@param*** *power  
 \*/* public void setPower(int power) {  
 if (power > 0) this.power = power;  
 else {  
 throw new IllegalArgumentException("Power <= 0 ! Set a new power.");  
 }  
 }  
  
 */\*\*  
 \** ***@return*** *name of Brand  
 \*/* public String getBrand() {  
 return brand;  
 }  
  
 */\*\*  
 \* Sets a new name of brand if it is  
 \*  
 \** ***@param*** *brand  
 \*/* public void setBrand(String brand) {  
 if (brand == "" || brand == null) this.brand = "\"Unknown brand\"";  
 else this.brand = brand;  
 }  
  
 */\*\*  
 \** ***@return*** *number of cylinders  
 \*/* public int getCylinder() {  
 return cylinder;  
 }  
  
 */\*\*  
 \* Sets a number of cylinders if it is more than 0.  
 \*  
 \** ***@param*** *cylinder  
 \*/* public void setCylinder(int cylinder) {  
  
 if (cylinder > 0) this.cylinder = cylinder;  
 else {  
 throw new IllegalArgumentException("Cylinder <= 0 ! Set a new cylinder.");  
 }  
 }  
  
  
}

Клас Track

package task31;  
  
import java.util.Objects;  
  
*/\*\*  
 \* Class for storing data of Track. Subclass of Car.  
 \*  
 \* <b>capacity</b>  
 \*  
 \** ***@author*** *Denis  
 \** ***@version*** *1.0  
 \*/*public class Track extends Car implements Drivable {  
 */\*\* capacity of track \*/* private int capacity;  
  
 */\*\*  
 \* Constructor with capacity  
 \*  
 \** ***@param*** *power  
 \** ***@param*** *brand  
 \** ***@param*** *cylinder  
 \** ***@param*** *capacity  
 \*/* public Track(int power, String brand, int cylinder, int capacity) {  
 super(power, brand, cylinder);  
 this.capacity = capacity;  
 }  
  
 */\*\*  
 \* Constructor without capacity  
 \*  
 \** ***@param*** *power  
 \** ***@param*** *brand  
 \** ***@param*** *cylinder  
 \*/* public Track(int power, String brand, int cylinder) {  
 super(power, brand, cylinder);  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *capacity of track  
 \*/* public int getCapacity() {  
 return capacity;  
 }  
  
 */\*\*  
 \* Sets a new capacity of track  
 \** ***@param*** *capacity  
 \*/* public void setCapacity(int capacity) {  
 if (capacity > 0) this.capacity = capacity;  
 else {  
 throw new IllegalArgumentException("Capacity <= 0 ! Set a new capacity.");  
 }  
  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *clone of Track  
 \*/* @Override  
 public Track clone() {  
 return new Track(getPower(), getBrand(), getCylinder(), getCapacity());  
 }  
  
 */\*\*  
 \* Compare two Tracks.  
 \*  
 \** ***@param*** *o  
 \** ***@return*** *if they are equal - true , else false.  
 \*/* @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Track)) return false;  
 Track track = (Track) o;  
 return getCapacity() == track.getCapacity() &&  
 getPower() == track.getPower() &&  
 getCylinder() == track.getCylinder() &&  
 Objects.*equals*(getBrand(), track.getBrand());  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *hashCode of track  
 \*/* @Override  
 public int hashCode() {  
 return Objects.*hash*(getCapacity(), getPower(), getBrand(), getCylinder());  
 }  
  
 */\*\*  
 \*  
 \** ***@return*** *string with all information.  
 \*/* @Override  
 public String toString() {  
 return "Track{" +  
 "power=" + getPower() +  
 ", brand='" + getBrand() + '\'' +  
 ", cylinder=" + getCylinder() +  
 ", capacity=" + capacity +  
 '}';  
 }  
}

Інтерфейс Drivable

package task31;  
  
public interface Drivable {  
  
 int getPower();  
  
 void setPower(int power);  
  
 String getBrand();  
  
 void setBrand(String brand);  
  
 int getCylinder();  
  
 void setCylinder(int cylinder);  
  
 int getCapacity();  
  
 void setCapacity(int capacity);  
  
}

Клас Main

package task31;  
  
public class Main {  
 public static void main(String[] args) {  
 Drivable track = new Track(300, null, 8, 3000);  
  
 System.*out*.println(track);  
  
 track.setPower(400);  
  
 track.setCylinder(6);  
  
 System.*out*.println(track);  
  
 Drivable t2 = new Track ( 400 , null , 6 );  
  
 t2.setCapacity(3000);  
  
 if (t2.equals(track)) System.*out*.println("They are equal!");  
  
  
 }  
  
}

Вивід

Track{power=300, brand='"Unknown brand"', cylinder=8, capacity=3000}

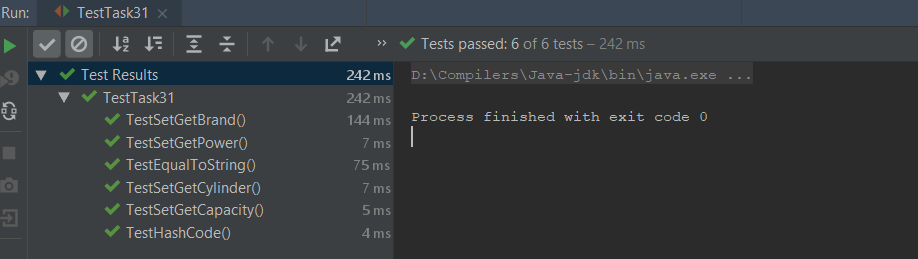
Track{power=400, brand='"Unknown brand"', cylinder=6, capacity=3000}

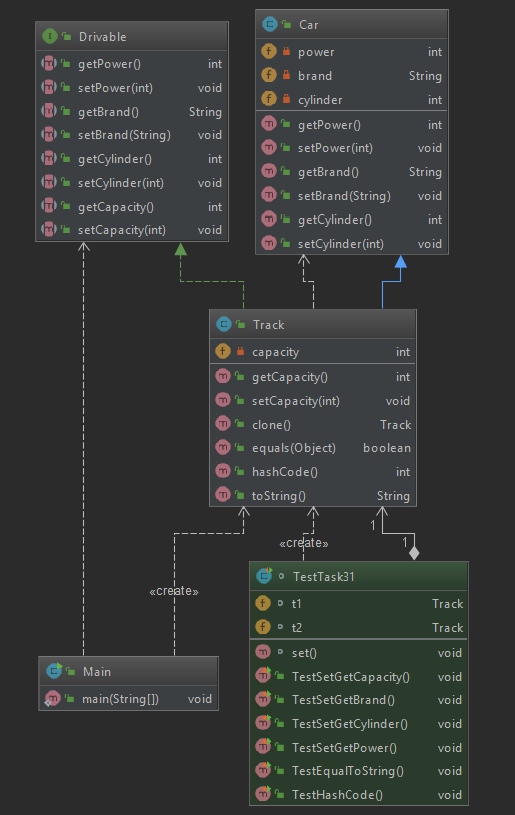
They are equal!

Тести до завдання

package task31;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class TestTask31 {  
  
 Track t1;  
 Track t2;  
  
 @BeforeEach  
 void set(){  
 t1 = new Track(300 , "" , 4 );  
 t2 = t1.clone();  
 }  
  
 @Test  
 void TestSetGetCapacity(){  
 t1.setCapacity(7000);  
 *assertEquals*(7000, t1.getCapacity());  
 }  
  
 @Test  
 void TestSetGetBrand(){  
 t1.setBrand("Toyota");  
 *assertEquals*("Toyota", t1.getBrand());  
 }  
  
 @Test  
 void TestSetGetCylinder(){  
 t1.setCylinder(6);  
 *assertEquals*(6, t1.getCylinder());  
 }  
  
 @Test  
 void TestSetGetPower(){  
 t1.setPower(200);  
 *assertEquals*(200, t1.getPower());  
 }  
  
 @Test  
 void TestEqualToString(){  
 *assertEquals*(true , t1.toString().equals(t2.toString()));  
 }  
  
 @Test  
 void TestHashCode(){  
 *assertEquals*(t1.hashCode(),t2.hashCode());  
 }  
  
}

Вивід



Діаграма