Jinan University

Java Programming Lab Report

Major: Computer Science & Technology
School: International School

Student name : _____
(p.s. your name on JNU academic system)

Student number : _____

Date of Submission (mm-dd-yyyy): _____

Instructor: Yuxia Sun

Table of Content

LAB 6 DATE: 5/9/2023

Student Name:_____ Student ID: _____

Problem 1. (10.1)

***10.1** (*The* `Time` *class*) Design a class named `Time`. The class contains:

- The data fields `hour`, `minute`, and `second` that represent a time.
- A no-arg constructor that creates a `Time` object for the current time. (The values of the data fields will represent the current time.)
- A constructor that constructs a `Time` object with a specified elapsed time since midnight, January 1, 1970, in milliseconds. (The values of the data fields will represent this time.)
- A constructor that constructs a `Time` object with the specified hour, minute, and second.
- Three getter methods for the data fields `hour`, `minute`, and `second`, respectively.
- A method named `setTime(long elapseTime)` that sets a new time for the object using the elapsed time. For example, if the elapsed time is `555550000` milliseconds, the hour is `10`, the minute is `19`, and the second is `10`.

Draw the UML diagram for the class then implement the class. Write a test program that creates three `Time` objects (using `new Time()`, `new Time(555550000)`, and `new Time(5, 23, 55)`) and displays their hour, minute, and second in the format hour:minute:second.

(*Hint*: The first two constructors will extract the hour, minute, and second from the elapsed time. For the no-arg constructor, the current time can be obtained using `System.currentTimeMillis()`, as shown in Listing 2.7, ShowCurrentTime.java. Assume the time is in GMT.)

**** Source Code / Solution :***

Time
-hour:int -minute:int -second
+Time():void +Time(long elapseTime):void +Time(int hour, int minute, int second):void +setTime(long elapseTime):void +displayTime():void

```java
public class Time {
    private int hour;
    private int minute;
    private int second;

    Time() {
        this.hour = (int) ((System.currentTimeMillis() / 1000 / 60 / 60) % 24);
        this.minute = (int) ((System.currentTimeMillis() / 1000 / 60) % 60);
        this.second = (int) ((System.currentTimeMillis() / 1000) % 60);
    }

    Time(long elapseTime){
        this.hour = (int) ((elapseTime / 1000 / 60 / 60) % 24);
        this.minute = (int) ((elapseTime / 1000 / 60) % 60);
        this.second = (int) ((elapseTime / 1000) % 60);
    }

    Time(int hour, int minute, int second){
        this.hour = hour;
        this.minute = minute;
        this.second = second;
```

```
    }

    public void setTime(long elapseTime){
        this.hour = (int) ((elapseTime / 1000 / 60 / 60) % 24);
        this.minute = (int) ((elapseTime / 1000 / 60) % 60);
        this.second = (int) ((elapseTime / 1000) % 60);
    }

    public void displayTime() {
        System.out.printf("Time: %02d:%02d:%02d GMT\n", this.hour, this.minute,
this.second);
    }
}
```

```
public class TimeTest {
    public static void main(String[] args) {
        Time test1 = new Time();
        System.out.println("Create the first Time object with no-arg constructor");
        test1.displayTime();

        Time test2 = new Time(555550000);
        System.out.println("Create the second Time object with specified elapsed time
constructor");
        test2.displayTime();

        Time test3 = new Time(5, 23, 55);
        System.out.println("Create the third Time object with specified hour, minute, and
second constructor");
        test3.displayTime();

        System.out.println("If we set the third time use System.currentTimeMillis(), it will
display:");
        test3.setTime(System.currentTimeMillis());
        test3.displayTime();
    }
}
```

**** Output:***

```
TimeTest
/Users/h3art/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/java
Create the first Time object with no-arg constructor
Time: 07:07:48 GMT
Create the second Time object with specified elapsed time constructor
Time: 10:19:10 GMT
Create the third Time object with  specified hour, minute, and second constructor
Time: 05:23:55 GMT
If we set the third time use System.currentTimeMillis(), it will display:
Time: 07:07:48 GMT

进程已结束,退出代码0
```

**** Debugging/Testing:***

Bug1: The hour, minute, and second algorithm for calculating time through the System.currentTimeMillis() method is wrong, and the result is wrong.
Fix: Recalculate to get the correct time calculation formula, and notice that the display format of time is GMT (Greenwich Mean Time) time.

Problem 2. (10.3)

10.3 (*The* **MyInteger** *class*) Design a class named **MyInteger**. The class contains:

- An **int** data field named **value** that stores the **int** value represented by this object.
- A constructor that creates a **MyInteger** object for the specified **int** value.
- A getter method that returns the **int** value.
- The methods **isEven()**, **isOdd()**, and **isPrime()** that return **true** if the value in this object is even, odd, or prime, respectively.
- The static methods **isEven(int)**, **isOdd(int)**, and **isPrime(int)** that return **true** if the specified value is even, odd, or prime, respectively.
- The static methods **isEven(MyInteger)**, **isOdd(MyInteger)**, and **isPrime(MyInteger)** that return **true** if the specified value is even, odd, or prime, respectively.
- The methods **equals(int)** and **equals(MyInteger)** that return **true** if the value in this object is equal to the specified value.
- A static method **parseInt(char[])** that converts an array of numeric characters to an **int** value.
- A static method **parseInt(String)** that converts a string into an **int** value.

Draw the UML diagram for the class then implement the class. Write a client program that tests all methods in the class.

**** Source Code / Solution :***

MyInteger
-value:int{readonly}
+getValue():int +isEven():boolean +isEven(int num):boolean +isEven(MyInteger myInteger):boolean +isOdd():boolean +isOdd(int num):boolean +isOdd(MyInteger myInteger):boolean +isPrime():boolean +isPrime(int num):boolean +isPrime(MyInteger myInteger):boolean +equals(int num):boolean +equals(MyInteger myInteger):boolean +parseInt(char[] chArray):int +parseInt(String string):int

```java
public class MyInteger {
    private final int value;

    MyInteger(int value) {
        this.value = value;
    }

    public int getValue() {
        return this.value;
    }

    public boolean isEven() {
        return this.value % 2 == 0;
    }

    public static boolean isEven(int num) {
        return num % 2 == 0;
    }
```

```java
public static boolean isEven(MyInteger myInteger) {
    return myInteger.getValue() % 2 == 0;
}

public boolean isOdd() {
    return this.value % 2 == 1;
}

public static boolean isOdd(int num) {
    return num % 2 == 1;
}

public static boolean isOdd(MyInteger myInteger) {
    return myInteger.getValue() % 2 == 1;
}

public boolean isPrime() {
    if (this.value < 2 || this.isEven() && this.value != 2) {
        return false;
    }

    int limit = (int) Math.sqrt(this.value);
    for (int i = 3; i <= limit; i += 2) {
        if (this.value % i == 0) {
            return false;
        }
    }
    return true;
}

public static boolean isPrime(int num){
    if (num < 2 || isEven(num) && num != 2) {
        return false;
    }

    int limit = (int) Math.sqrt(num);
    for (int i = 3; i <= limit; i += 2) {
        if (num % i == 0) {
```

```
            return false;
        }
    }
    return true;
}

public static boolean isPrime(MyInteger myInteger){
    if (myInteger.getValue() < 2 || isEven(myInteger) && myInteger.getValue() != 2) {
        return false;
    }

    int limit = (int) Math.sqrt(myInteger.getValue());
    int value = myInteger.getValue();
    for (int i = 3; i <= limit; i += 2) {
        if (value % i == 0) {
            return false;
        }
    }
    return true;
}

public boolean equals(int num){
    return this.value == num;
}

public boolean equals(MyInteger myInteger){
    return this.value == myInteger.getValue();
}

public static int parseInt(char[] chArray){
    int positiveOrNegative = 1, result = 0;
    boolean isFirstCharacter = true;

    for(char ch:chArray){
        if (ch == '-' && isFirstCharacter){
            positiveOrNegative = -1;
        }else if(ch >= '0' && ch <= '9'){
            result = (result << 3) + (result << 1) + (ch ^ 48);
        }else {
```

```
            throw new IllegalArgumentException("Illegal character array");
        }
        isFirstCharacter = false;
    }

    return result * positiveOrNegative;
  }

  public static int parseInt(String string){
    return parseInt(string.toCharArray());
  }
}
```

```
package lab6.chapter10;

import java.util.Scanner;

public class MyIntegerTest {
  public static boolean isPrime(int num) {
    if (num < 0 || (num % 2 == 0 && num != 2)) {
      return false;
    }
    for (int i = 3; i <= (int) Math.sqrt(num); i += 2) {
      if (num % i == 0) {
        return false;
      }
    }
    return true;
  }

  public static boolean objectTest(int testNum) {
    MyInteger myInteger = new MyInteger(testNum);
    System.out.println("Create a MyInteger object with value " + myInteger.getValue());

    if (myInteger.isEven() && MyInteger.isEven(myInteger) && myInteger.getValue() % 2 ==
0) {
      System.out.println(myInteger.getValue() + " is an even number");
    } else if (myInteger.isOdd() && MyInteger.isOdd(myInteger) && myInteger.getValue() %
2 == 1) {
      System.out.println(myInteger.getValue() + " is an odd number");
```

```
    } else {
        return false;
    }

    if (myInteger.isPrime() == MyInteger.isPrime(myInteger) &&
            myInteger.isPrime() == isPrime(myInteger.getValue())) {
        if (isPrime(myInteger.getValue())) {
            System.out.println(myInteger.getValue() + " is an prime number");
        } else {
            System.out.println(myInteger.getValue() + " isn't an prime number");
        }
    } else {
        return false;
    }
    return true;
}

public static boolean parseTest(int testNum) {
    // String parsing
    System.out.println("Parse a String: " + testNum);
    String testString = String.valueOf(testNum);

    if (MyInteger.parseInt(testString) == testNum) {
        System.out.println("Parse " + testNum + " successful");
    } else {
        System.out.println("Parse " + testNum + " failed");
        return false;
    }

    // char array parsing
    System.out.println("Parse a char array: " + testNum);
    char[] testCharArray = testString.toCharArray();
    if (MyInteger.parseInt(testCharArray) == testNum) {
        System.out.println("Parse " + testNum + " successful");
    } else {
        System.out.println("Parse " + testNum + " failed");
        return false;
    }
```

```
        return true;
    }

    public static boolean intTest(int testNum) {
        System.out.println("Create a int number with value " + testNum);

        if (MyInteger.isEven(testNum) && testNum % 2 == 0) {
            System.out.println(testNum + " is an even number");
        } else if (MyInteger.isOdd(testNum) && testNum % 2 == 1) {
            System.out.println(testNum + " is an odd number");
        } else {
            return false;
        }

        if (MyInteger.isPrime(testNum) == isPrime(testNum)) {
            if (isPrime(testNum)) {
                System.out.println(testNum + " is an prime number");
            } else {
                System.out.println(testNum + " isn't an prime number");
            }
        } else {
            return false;
        }
        return true;
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter an integer: ");
        int testNumber = input.nextInt();

        if (objectTest(testNumber)) {
            System.out.println("\nObject test: Success\n");
        } else {
            throw new RuntimeException("Object test: Fail");
        }

        if (intTest(testNumber)) {
            System.out.println("\nint test: Success\n");
```

```
        } else {
            throw new RuntimeException("int test: Fail");
        }

        if (parseTest(testNumber)) {
            System.out.println("\nParse test: Success\n");
        } else {
            throw new RuntimeException("Parse test: Fail");
        }
    }
}
```

**** Output:***

```
MyIntegerTest
/Users/h3art/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/java
Enter an integer: 12
Create a MyInteger object with value 12
12 is an even number
12 isn't an prime number

Object test: Success

Create a int number with value 12
12 is an even number
12 isn't an prime number

int test: Success

Parse a String: 12
Parse 12 successful
Parse a char array: 12
Parse 12 successful

Parse test: Success


进程已结束,退出代码0
```

**** Debugging/Testing:***

Bug1: The method to realize the prime number judgment was wrong, and 2 was included in the composite number.
Fix: Modify the judgment condition to start looking for prime numbers from 3 with a step size of 2.

Problem 3. (10.7)

****10.7** (*Game: ATM machine*) Use the **Account** class created in Programming Exercise 9.7 to simulate an ATM machine. Create 10 accounts in an array with id **0**, **1**, . . . , **9**, and an initial balance of $100. The system prompts the user to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, the main menu is displayed as shown in the sample run. You can enter choice **1** for viewing the current balance, **2** for withdrawing money, **3** for depositing money, and **4** for exiting the main menu. Once you exit, the system will prompt for an id again. Thus, once the system starts, it will not stop.

**** Source Code / Solution :***

```java
import java.util.Date;

public class Account {
   private int id;
   private double balance;
   // Note: annualInterestRate is a percentage
   private static double annualInterestRate = 0;
   private final Date dateCreated;

   Account() {
      this.id = 0;
      this.balance = 0;
      this.dateCreated = new Date();
   }

   Account(int id, double balance) {
      this.id = id;
      this.balance = balance;
      this.dateCreated = new Date();
   }

   public int getId() {
      return this.id;
   }

   public void setId(int id) {
      this.id = id;
   }
```

```java
public double getBalance() {
    return this.balance;
}

public void setBalance(double balance) {
    this.balance = balance;
}

public static double getAnnualInterestRate() {
    return annualInterestRate / 100;
}

public static void setAnnualInterestRate(double newAnnualInterestRate) {
    annualInterestRate = newAnnualInterestRate;
}

public Date getDateCreated() {
    return this.dateCreated;
}

public static double getMonthlyInterestRate() {
    return annualInterestRate / 100 / 12;
}

public double getMonthlyInterest() {
    return this.balance * getMonthlyInterestRate();
}

public boolean withdraw(double withdrawBalance) {
    if (this.balance - withdrawBalance >= 0) {
        this.balance -= withdrawBalance;
        return true;
    } else {
        return false;
    }
}

public double deposit(double depositBalance){
```

```
        this.balance += depositBalance;
        return this.balance;
    }
}
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class ATMMachine {
    private int ACCOUNTS_NUM;
    private List<Account> accountList;

    ATMMachine(int accountsNumber) {
        ACCOUNTS_NUM = accountsNumber;
        accountList = new ArrayList<>();
        for (int i = 0; i < ACCOUNTS_NUM; i++) {
            accountList.add(new Account(i, 100));
        }
    }

    public Account userIDMatch(int ID) {
        for (Account account : accountList) {
            if (ID == account.getId()) {
                return account;
            }
        }
        System.out.println("Wrong ID, try again");
        return null;
    }

    public boolean mainMenu(Account account, Scanner input) {
        switch (input.nextInt()) {
            case 1:
                System.out.println("The balance is " + account.getBalance());
                break;
            case 2:
```

```
            System.out.print("Enter an amount to withdraw: ");
            if (!account.withdraw(input.nextDouble())) {
                System.out.println("Insufficient balance, withdrawal failed");
            }
            break;
        case 3:
            System.out.print("Enter an amount to deposit: ");
            account.deposit(input.nextDouble());
            break;
        case 4:
            System.out.println();
            return false;
        default:
            System.out.println("Wrong choice, try again");
    }
    System.out.println();
    return true;
}

public static void main(String[] args) {
    ATMMachine machine = new ATMMachine(10);
    Scanner input = new Scanner(System.in);
    Account user;

    while (true) {
        do {
            System.out.print("Enter an ID: ");
            user = machine.userIDMatch(input.nextInt());
        } while (user == null);

        do {
            System.out.print("Main menu\n" +
                    "1: check balance\n" +
                    "2: withdraw\n" +
                    "3: deposit\n" +
                    "4: exit\n" +
                    "Enter a choice: ");
        } while (machine.mainMenu(user, input));
    }
```

```
    }
}
```

**** Output:***

```
ATMMachine ×
/Users/h3art/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/java
Enter an ID: 4
Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 1
The balance is 100.0

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 2
Enter an amount to withdraw: 3

ATMMachine ×
Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 1
The balance is 97.0

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 3
Enter an amount to deposit: 10

Main menu
1: check balance
2: withdraw
3: deposit
4: exit
Enter a choice: 1
The balance is 107.0
```

s

**** Debugging/Testing:***

```
Bug1: Error in profit calculation of Account class(Although
it is no used).
Fix: Modify profit calculation formula.
Bug2: The sequential execution flow of the ATM program is
wrong, resulting in an infinite loop in nested while(true)
loop.
Fix: Use the debugger to locate sequentially executed bugs
and modify the program.
```

Problem 4. (10.9)(Optional)

****10.9** (*The* **Course** *class*) Revise the **Course** class as follows:

- Revise the **getStudents()** method to return an array whose length is the same as the number of students in the course. (*Hint*: create a new array and copy students to it.)
- The array size is fixed in Listing 10.6. Revise the **addStudent** method to automatically increase the array size if there is no room to add more students. This is done by creating a new larger array and copying the contents of the current array to it.
- Implement the **dropStudent** method.
- Add a new method named **clear()** that removes all students from the course.

Test your program using https://liveexample.pearsoncmg.com/test/Exercise10_09.txt

**** Source Code / Solution :***

```java
public class Course {
  private int capacity = 100;
  private String courseName;
  private String[] students = new String[capacity];
  private int numberOfStudents;

  public Course(String courseName) {
    this.courseName = courseName;
  }

  public void addStudent(String student) {
    if (numberOfStudents + 1 > capacity) {
      capacity *= 2;
      String[] increasedStudents = new String[capacity];
      System.arraycopy(students, 0, increasedStudents, 0, numberOfStudents);
```

```
        students = increasedStudents;
    }
    students[numberOfStudents] = student;
    numberOfStudents++;

}

public String[] getStudents() {
    String[] exactStudents = new String[numberOfStudents];
    System.arraycopy(students, 0, exactStudents, 0, numberOfStudents);
    return exactStudents;
}

public int getNumberOfStudents() {
    return numberOfStudents;
}

public String getCourseName() {
    return courseName;
}

public void dropStudent(String student) {
    // Left as an exercise in Programming Exercise 10.9
    // Search
    int index = -1;
    for (int i = 0; i < numberOfStudents; i++) {
        if (students[i].equals(student)) {
            index = i;
            break;
        }
    }

    // Drop
    if (index == -1) {
        System.out.println("Student: " + student + " doesn't exist");
    } else {
        for (int i = index; i < numberOfStudents; i++) {
            students[i] = students[i + 1];
        }
```

```
            numberOfStudents--;
        }

        // Capacity adjustment
        if(numberOfStudents <= capacity / 2){
            capacity /= 2;
            String[] increasedStudents = new String[capacity];
            System.arraycopy(students, 0, increasedStudents, 0, numberOfStudents);
            students = increasedStudents;
        }
    }

    public void clear() {
        numberOfStudents = 0;
        capacity = 100;
        students = new String[capacity];
    }
}
```

```
public class CourseTest {
    public static void main(String[] args) {
        Course course1 = new Course("Data Structures");
        Course course2 = new Course("Database Systems");

        course1.addStudent("Peter Jones");
        course1.addStudent("Brian Smith");
        course1.addStudent("Anne Kennedy");
        course1.addStudent("Susan Kennedy");
        course1.addStudent("John Kennedy");
        course1.addStudent("Kim Johnson");
        course1.addStudent("S1");
        course1.addStudent("S2");
        course1.addStudent("S3");
        course1.addStudent("S4");
        course1.addStudent("S5");
        course1.addStudent("S6");
        course1.addStudent("S7");

        course2.addStudent("Peter Jones");
        course2.addStudent("Steve Smith");
```

```java
        System.out.println("Number of students in course1: "
                + course1.getNumberOfStudents());
        String[] students = course1.getStudents();
        for (int i = 0; i < students.length; i++)
            System.out.print(students[i] + ", ");

        System.out.println();
        System.out.println("Number of students in course2: "
                + course2.getNumberOfStudents());

        course1.dropStudent("S1");
        System.out.println("Number of students in course1: "
                + course1.getNumberOfStudents());
        students = course1.getStudents();
        for (int i = 0; i < course1.getNumberOfStudents(); i++)
            System.out.print(students[i] + (i < course1.getNumberOfStudents() - 1 ? ", " : " "));

        course1.clear();
        System.out.println("\nNumber of students in course1: "
                + course1.getNumberOfStudents());
    }
}
```

**** Output:***

```
CourseTest
/Users/h3art/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/java ...
Number of students in course1: 13
Peter Jones, Brian Smith, Anne Kennedy, Susan Kennedy, John Kennedy, Kim Johnson, S1, S2, S3, S4, S5, S6, S7,
Number of students in course2: 2
Number of students in course1: 12
Peter Jones, Brian Smith, Anne Kennedy, Susan Kennedy, John Kennedy, Kim Johnson, S2, S3, S4, S5, S6, S7
Number of students in course1: 0

进程已结束,退出代码0
```

**** Debugging/Testing:***

Bug1: An error occurred in copying the array of the original storage students.
Fix: I found the System.arraycopy() method provided by Java, modified the program and copied the original student array correctly.

Problem 5. (10.13)(Optional)

***10.13** (*Geometry: the* `MyRectangle2D` *class*) Define the `MyRectangle2D` class that contains:

- Two `double` data fields named `x` and `y` that specify the center of the rectangle with getter and setter methods. (Assume the rectangle sides are parallel to `x-` or `y-`axis.)
- The data fields `width` and `height` with getter and setter methods.
- A no-arg constructor that creates a default rectangle with (`0`, `0`) for (`x`, `y`) and `1` for both `width` and `height`.
- A constructor that creates a rectangle with the specified `x`, `y`, `width`, and `height`.
- A method `getArea()` that returns the area of the rectangle.
- A method `getPerimeter()` that returns the perimeter of the rectangle.
- A method `contains(double x, double y)` that returns `true` if the specified point (`x`, `y`) is inside this rectangle (see Figure 10.24a).
- A method `contains(MyRectangle2D r)` that returns `true` if the specified rectangle is inside this rectangle (see Figure 10.24b).
- A method `overlaps(MyRectangle2D r)` that returns `true` if the specified rectangle overlaps with this rectangle (see Figure 10.24c).

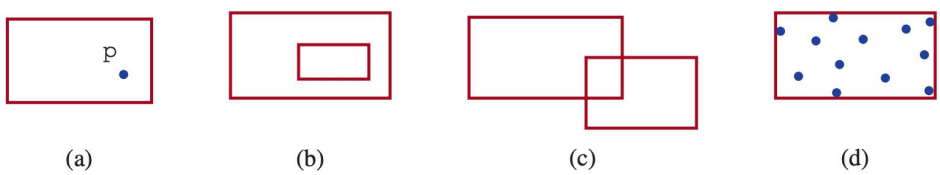

FIGURE 10.24 (a) A point is inside the rectangle. (b) A rectangle is inside another rectangle. (c) A rectangle overlaps another rectangle. (d) Points are enclosed inside a rectangle.

Draw the UML diagram for the class then implement the class. Write a test program that creates a `MyRectangle2D` object `r1` (`new MyRectangle2D (2, 2, 5.5, 4.9)`), displays its area and perimeter, and displays the result of `r1.contains(3, 3)`, `r1.contains(new MyRectangle2D(4, 5, 10.5, 3.2))`, and `r1.overlaps(new MyRectangle2D(3, 5, 2.3, 5.4))`.

**** Source Code / Solution :***

MyRectangle2D
-x:double -y:double -width:double -height:double
+MyRectangle2D():void +MyRectangle2D(double x, double y, double width, double height):void +getX():double +setX():void +getY():double +setY():void +getWidth():double +setWidth():void +getHeight():double +setHeight():void +getArea():double +getPerimeter():double +contains(double x, double y):boolean +contains(MyRectangle2D r):boolean +overlaps(MyRectangle2D r):boolean

```java
public class MyRectangle2D {
    private double x;
    private double y;
    private double width;
    private double height;

    MyRectangle2D() {
        this.x = 0;
        this.y = 0;
        this.width = 1;
        this.height = 1;
    }

    MyRectangle2D(double x, double y, double width, double height) {
        this.x = x;
        this.y = y;
        this.width = width;
```

```
    this.height = height;
}

public boolean contains(double x, double y) {
    return x < this.x + width / 2 &&
            x > this.x - width / 2 &&
            y < this.y + height / 2 &&
            y > this.y - height / 2;
}

public boolean contains(MyRectangle2D r) {
    return r.getX() + r.getWidth() / 2 < this.x + width / 2 &&
            r.getX() - r.getWidth() / 2 > this.x - width / 2 &&
            r.getY() + r.getHeight() / 2 < this.y + height / 2 &&
            r.getY() - r.getHeight() / 2 > this.y - height / 2;
}

public boolean overlaps(MyRectangle2D r) {
    return contains(r.getX() + r.getWidth() / 2, r.getY() + r.getHeight() / 2) ||
            contains(r.getX() + r.getWidth() / 2, r.getY() - r.getHeight() / 2) ||
            contains(r.getX() - r.getWidth() / 2, r.getY() + r.getHeight() / 2) ||
            contains(r.getX() - r.getWidth() / 2, r.getY() - r.getHeight() / 2);
}

public double getArea() {
    return width * height;
}

public double getPerimeter() {
    return 2 * (width + height);
}

public void setX(double x) {
    this.x = x;
}

public void setY(double y) {
    this.y = y;
}
```

```
    public void setWidth(double width) {
        this.width = width;
    }

    public void setHeight(double height) {
        this.height = height;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }

    public double getWidth() {
        return width;
    }

    public double getHeight() {
        return height;
    }
}
```

```
public class MyRectangle2DTest {
    public static void main(String[] args) {
        MyRectangle2D r1 = new MyRectangle2D(2, 2, 5.5, 4.9);

        System.out.println("The area of r1 is " + r1.getArea() + " and its perimeter is " +
r1.getPerimeter());

        System.out.println("r1 contains point(3, 3): " + r1.contains(3, 3));

        System.out.println("r1 contains MyRectangle2D(4, 5, 10.5, 3.2): " +
            r1.contains(new MyRectangle2D(4, 5, 10.5, 3.2)));

        System.out.println("r1 overlaps MyRectangle2D(3, 5, 2.3, 5.4): " +
            r1.overlaps(new MyRectangle2D(3, 5, 2.3, 5.4)));
```

```
    }
}
```

**** Output:***

```
MyRectangle2DTest
/Users/h3art/Library/Java/JavaVirtualMachines/jdk1.8.0_202.jdk/Contents/Home/bin/java
The area of r1 is 26.950000000000003 and its perimeter is 20.8
r1 contains point(3, 3): true
r1 contains MyRectangle2D(4, 5, 10.5, 3.2): false
r1 overlaps MyRectangle2D(3, 5, 2.3, 5.4): true

进程已结束,退出代码0
```

**** Debugging/Testing:***

Bug1: The contains() method is designed incorrectly, and there are situations that have not been considered.
Fix: After formatting the code, consider every situation of the contains() method clearly, and reimplement the contains() method.