Software Testing - HW01

Triangle/Person

資訊三丙
D1009331 吳克廷

目錄

前言

本文件為解釋程式碼的運作原理，因此會擷取片段的程式碼，並非完整的程式碼。若想參閱完整的程式架構，可以根據以下所提供的網址，前往GitHub進行參閱，老師謝謝。

- Main.java: https://github.com/alecwu44743/Software-Testing/blob/master/HW01/src/main/java/org/example/Main.java
- Person.java: https://github.com/alecwu44743/Software-Testing/blob/master/HW01/src/main/java/org/example/Person.java

- MainTest.java: https://github.com/alecwu44743/Software-Testing/blob/master/HW01/src/test/java/org/example/MainTest.java
- PersonTest.java: https://github.com/alecwu44743/Software-Testing/blob/master/HW01/src/test/java/org/example/PersonTest.java

Triangle - 設計

在Triangle中，將三邊定義為a, b, c，讓使用者分別輸入，而本程式碼運用了try-catch，可以避免使用在輸入錯誤時，而發生程式碼直接終止的情況。

```java
// input side a
while (true) {
    System.out.print("Enter side a: ");
    try {
        a = Double.parseDouble(scanner.next());
        numberCheck(a); // check if a is valid
        break; // will only get to here if input was a double
    } catch (NumberFormatException ignore) {
        System.out.println("Invalid input");
    }catch (Exception e){
        isValid = false;
        System.out.println(e.getMessage());
    }
}
```

```java
// input side b
while (true) {
    System.out.print("Enter side b: ");
    try {
        b = Double.parseDouble(scanner.next());
        numberCheck(b); // check if b is valid
        break; // will only get to here if input was a double
    } catch (NumberFormatException ignore) {
        System.out.println("Invalid input");
    }catch (Exception e){
        isValid = false;
        System.out.println(e.getMessage());
    }
}
```

```java
// input side c
while (true) {
    System.out.print("Enter side c: ");
    try {
        c = Double.parseDouble(scanner.next());
        numberCheck(c); // check if c is valid
        break; // will only get to here if input was a double
    } catch (NumberFormatException ignore) {
        System.out.println("Invalid input");
    }catch (Exception e){
        isValid = false;
        System.out.println(e.getMessage());
    }
}
```

如上三張圖示，利用try-catch，可以讓使用者一直輸入，直到輸入正確後才進入到下一個狀態，而發生exception會另外處理。前頁有提到exception，而這裡會判斷是否為數字，使用者即為可能無意輸入到英文字母或符號等，因此這裡會提醒使用者”Invalid input”。

```java
public static void numberCheck(double num) throws Exception {
    if (num < 0) {
        throw new Exception("Side a cannot be negative");
    }
    else if (num == 0) {
        throw new Exception("Side a cannot be zero");
    }
}
```

另外也開發了numberCheck()，如上圖，這個函式可以依據使用者的錯誤類別，而產出不同的exception，若使用者所輸入的數字小於0，則會輸出“邊長不可為負的”，若是0，則輸出”邊長不可為0”，讓使用者可以快速地意識到在輸入上的失誤。下個階段，將討論checkTriangle()。

當使用者輸入無誤後，可進入判斷三角形的環節，如下圖，將a, b, c三個邊分別傳入，先一開始判斷三角形是否成立，因此這裡用邊長作為判斷，若判斷不能構成三角形，將回傳0，第二種情況是若三邊皆等長，那麼為正三角形，將回傳1，第三種情況為其種兩邊相等，因此為等腰三角形，將回傳2，最後情況為以上條件皆不相等，因此為一般的三角形，回傳3。

```java
// check if a triangle is equilateral, isosceles, or scalene
int checkTriangle(double a, double b, double c) {
    if (a + b <= c || b + c <= a || a + c <= b) {
        return 0; // not a triangle
    }
    else if (a == b && b == c) {
        return 1; // equilateral triangle
    } else if (a == b || b == c || a == c) {
        return 2; // isosceles triangle
    } else {
        return 3; // scalene triangle
    }
}
```

最後，判斷為後將輸出三角形的型態，以及三邊長等資訊，程式如下圖。

```java
System.out.println("The three sides of the triangle are " + a + ", " + b + ", and " + c);

// check the type of triangle, and print the result
Main main = new Main();
int result = main.checkTriangle(a, b, c);
if (result == 0) {
    System.out.println("This is not a triangle");
} // not a triangle (sum of two sides <= third side)
else if (result == 1) {
    System.out.println("This is an equilateral triangle");
} else if (result == 2) {
    System.out.println("This is an isosceles triangle");
} else {
    System.out.println("This is a scalene triangle");
}
```

Triangle - 程式執行結果

- 正常輸入

```
/Library/Java/JavaVirtualMachines/adoptopenjdk-16.jdk/Contents/Home/bin/java
Triangle Checker
Enter side a: 6
Enter side b: 6
Enter side c: 2
The three sides of the triangle are 6.0, 6.0, and 2.0
This is an isosceles triangle

Process finished with exit code 0
```

- 發生錯誤輸入

```
/Library/Java/JavaVirtualMachines/adoptopenjdk-16.jdk/Contents/Home/bin/java
Triangle Checker
Enter side a: a
Invalid input
Enter side a: 6
Enter side b: 6
Enter side c: 6
The three sides of the triangle are 6.0, 6.0, and 6.0
This is an equilateral triangle
```

因此這裡會提醒使用者發生錯誤。

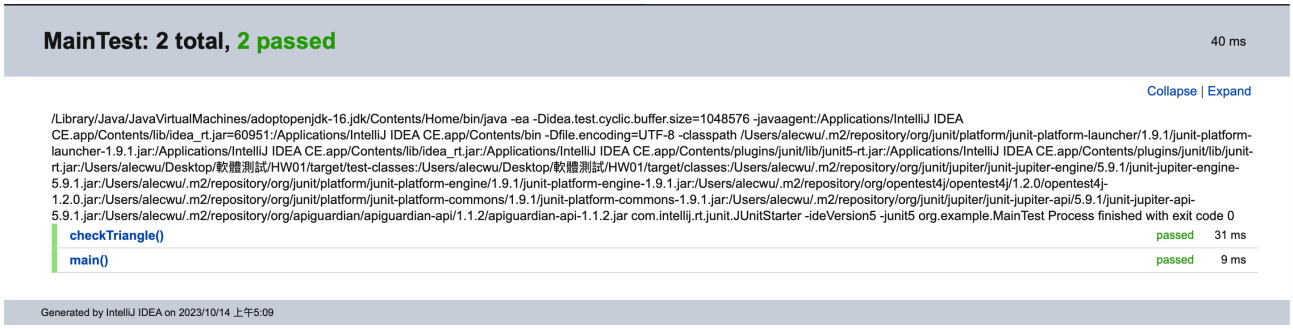
- 三角形邊長不成立

```
/Library/Java/JavaVirtualMachines/adoptopenjdk-16.jdk/Contents/Home/bin/java
Triangle Checker
Enter side a: 1
Enter side b: 2
Enter side c: 3
The three sides of the triangle are 1.0, 2.0, and 3.0
This is not a triangle
```

這裡發生了三角形不成立的狀況，因此會提醒使用者

Triangle - Unit Test

- 測試報告書



這裡進行了2種測試，而測試結果為全部通過，下個階段將介紹每個測試的詳細過程與設計。

- 測試前初始化

```
1       Main main = new Main();
2
3       @BeforeEach // To initialize the test
4       void init() {
5           main = new Main();
6   //        System.out.println("init");
7           System.out.println("[>] Init MainTest\n");
8       }
```

為了確保每個測試的情況皆保持乾淨，因此會於開始進行每個階段的測試前，進行初始化，並且提供訊息已提示開發者。

- CheckTriangle

```
checkTriangle()                                                    passed  31 ms
    [>] Init MainTest
    [+] Here is checkTriangle test
    0 -> not a triangle
    1 -> equilateral triangle
    2 -> isosceles triangle
    3 -> scalene triangle
    [v] checkTriangle test passed
```

由測試報告書所提供的輸出結果，可以看到這裡的設計會有[+] [v]等讓開發者好辨認，以及告知開發者每個代號分別代表者何種三角形，並且對每個種類的三角形進行測試，程式碼如下圖。

```java
@org.junit.jupiter.api.Test
void checkTriangle() {
    System.out.println("[+] Here is checkTriangle test");
    System.out.println("0 -> not a triangle");
    System.out.println("1 -> equilateral triangle");
    System.out.println("2 -> isosceles triangle");
    System.out.println("3 -> scalene triangle");

    int actualValue = main.checkTriangle(6, 6, 3);
    int expectedValue = 2;

    assertEquals(expectedValue, actualValue);

    assertEquals(1, main.checkTriangle(6, 6, 6), "Should be equilateral triangle");
    assertEquals(2, main.checkTriangle(6, 6, 3), "Should be isosceles triangle");
    assertEquals(3, main.checkTriangle(6, 5, 3), "Should be scalene triangle");
    assertEquals(3, main.checkTriangle(6, 3, 5), "Should be scalene triangle");
    assertEquals(1, main.checkTriangle(6, 6, 6), "Should be equilateral triangle");
    assertEquals(2, main.checkTriangle(6, 6, 3), "Should be isosceles triangle");
    assertEquals(0, main.checkTriangle(6, 6, 12), "Should be not a triangle");

    System.out.println("[v] checkTriangle test passed\n");
}
```

- main

```
main()                                                             passed  9 ms
    [>] Init MainTest
    [+] Here is main test for exception
    Case #1: pass
    Input is -1, Exception message: Side a cannot be negative
    Case #2: pass
    Input is 0, Exception message: Side a cannot be zero
    [v] main test passed
```

這裡主要在測試使用者在輸入時，當錯誤輸入時，exception是否可以正常運作，因此這裡給了0, -1，詳細程式碼請見下頁。

```java
@org.junit.jupiter.api.Test
void main() {
    System.out.println("[+] Here is main test for exception");

    Exception exception = assertThrows(Exception.class, () -> {
        main.numberCheck(-1);
    });
    System.out.println("Case #1: pass");
    System.out.println("Input is -1, " + "Exception message: " + exception.getMessage());

    exception = assertThrows(Exception.class, () -> {
        main.numberCheck(0);
    });
    System.out.println("Case #2: pass");
    System.out.println("Input is 0, " + "Exception message: " + exception.getMessage());

    System.out.println("[v] main test passed");
}
```

這裡運用了assertThrows，測是每個exception的情況，確認每個錯誤的輸入時，是否可以產生出相對應且正常的exception。

最後，每筆測試都如期通過，也代表程式在未來可以正常運作。

Person - 設計

Person的設計主要是希望使用者可以輸入名稱，年月日的生日、身高及體重，最後紀錄BMI，因此以下是這次設計所使用到的屬性。

```java
public class Person {
    String name;
    int yearOfBirth;
    int monthOfBirth;
    int dayOfBirth;
    int age;
    double height;
    double weight;
    double bmi;

    public Person(String _name, int _yearOfBirth, int _monthOfBirth, int _dayOfBirth, double _height, double _weight) {
        this.name = _name;
        this.yearOfBirth = _yearOfBirth;
        this.monthOfBirth = _monthOfBirth;
        this.dayOfBirth = _dayOfBirth;

        LocalDate currentDate = LocalDate.now();
        this.age = currentDate.getYear() - this.yearOfBirth;

        this.height = _height;
        this.weight = _weight;
        this.bmi = calculateBMI();
    }
}
```

使用者在經由輸入後，會進行建構，而年齡的部分是使用java內奸的時間package，直接取得即時的年份再做計算，最後在使用內建的calculateBMI計算BMI，而輸入的處理會於下個階段進行介紹。

- 輸入控制

```java
    public static void main(String[] args) {
        String _name;
        int _yearOfBirth, _monthOfBirth, _dayOfBirth;
        double _height, _weight;

        Scanner scanner = new Scanner(System.in);


        while (true) {
            try {
                System.out.print("Enter name: ");
                _name = scanner.nextLine();
                nameCheck(_name);
                break;
            } catch (Exception ignore) {
                System.out.println("Invalid input");
            }
        }

        while (true) {
            try {
                System.out.print("Enter year of birth: ");
                _yearOfBirth  = Integer.parseInt(scanner.nextLine());
                yearCheck(_yearOfBirth);
                break;
            } catch (NumberFormatException ignore) {
                System.out.println("Invalid input");
            }catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }

        while (true) {
            try {
                System.out.print("Enter month of birth: ");
                _monthOfBirth = Integer.parseInt(scanner.nextLine());
                monthCheck(_monthOfBirth, _yearOfBirth);
                break;
            } catch (NumberFormatException ignore) {
                System.out.println("Invalid input");
            }catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }

        while (true) {
            try {
                System.out.print("Enter day of birth: ");
                _dayOfBirth = Integer.parseInt(scanner.nextLine());
                dayCheck(_monthOfBirth, _dayOfBirth, _yearOfBirth);
                break;
            } catch (NumberFormatException ignore) {
                System.out.println("Invalid input");
            }catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }

        while (true) {
            try {
                System.out.print("Enter height(cm): ");
                _height = Double.parseDouble(scanner.next());
                heightCheck(_height);
                break;
            } catch (NumberFormatException ignore) {
                System.out.println("Invalid input");
            }catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }

        while (true) {
            try {
                System.out.print("Enter weight(kg): ");
                _weight = Double.parseDouble(scanner.next());
                weightCheck(_weight);
                break;
            } catch (NumberFormatException ignore) {
                System.out.println("Invalid input");
            }catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }

        Person person = new Person(_name, (int)_yearOfBirth, (int)_monthOfBirth, (int)_dayOfBirth, _height, _weight);

        System.out.println(person.showMyInfo());
    }
}

```

在這裡一樣運用了try-catch的方法，讓使用者在輸入時可以增加容錯入，而對於每個輸入也進行更進一步的exception處理。

```java
public static void nameCheck(String name) throws Exception {
    if (name.length() < 1 || name.length() > 100) {
        throw new Exception("Name cannot be less than 1 or greater than 100 characters");
    }
}

public static void yearCheck(int year) throws Exception {
    LocalDate currentDate = LocalDate.now();
    if (year < 1850 || year > currentDate.getYear()) {
        throw new Exception("Year of birth cannot be less than 1850 or greater than current year " + currentDate.getYear());
    }
}

public static void monthCheck(int month, int year) throws Exception {
    LocalDate currentDate = LocalDate.now();
    if (month < 1 || month > 12) {
        throw new Exception("Month of birth cannot be less than 1 or greater than 12");
    }
    else{
        if (year == currentDate.getYear()) {
            if (month > currentDate.getMonthValue()) {
                throw new Exception("Month of birth cannot be greater than current month");
            }
        }
    }
}

public static boolean isLeapYear(int year) {
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}
```

- nameCheck(): 用於判斷字串長度是否於指定的範圍內。
- yearCheck(): 藉由取的目前的年份，不僅可以控制在範圍內，也可以根據當前的時間，避免使用者輸入及不合理的年份。
- monthCheck(): 可以確定是否於一至十二月，並且也可以判對時間是否與當前的時間超前。
- isLeapYear(): 由於年、月、日與閏年的變化十分有關係，因此藉由此函式可以判斷是否為閏年，進而取得更準確的判斷。

```java
public static void dayCheck(int month, int day, int year) throws Exception {
    LocalDate currentDate = LocalDate.now();
    if (day < 1 || day > 31) {
        throw new Exception("Day of birth cannot be less than 1 or greater than 31");
    }
    else{
        if (year == currentDate.getYear()) { // check if year of birth is current year
            if (month == currentDate.getMonthValue()) {
                if (day > currentDate.getDayOfMonth()) {
                    throw new Exception("Day of birth cannot be greater than current day");
                }
            }
        }
        else{
            if (month == 2) { // check if month of birth is February
                if (isLeapYear(year)) {
                    if (day > 29) { // if leap year
                        throw new Exception("Day of birth cannot be greater than 29");
                    }
                }
                else{
                    if (day > 28) { // if not leap year
                        throw new Exception("Day of birth cannot be greater than 28");
                    }
                }
            }
            else if (month == 4 || month == 6 || month == 9 || month == 11) { // check if month of birth is April, June, September, or November
                if (day > 30) {
                    throw new Exception("Day of birth cannot be greater than 30");
                }
            }
        }
    }
}

public static void heightCheck(double height) throws Exception {
    if (height < 10 || height > 300) {
        throw new Exception("Height cannot be less than 0 or greater than 300\n" + "Please use cm");
    }
}

public static void weightCheck(double weight) throws Exception {
    if (weight <= 0 || weight > 500) {
        throw new Exception("Weight cannot be less than 0 or greater than 500\n" + "Please use kg");
    }
}
```

- dayCheck(): 日期的判斷較為複雜，這裡運用了閏年的判斷來確定當年當月的日最大值，其中也包括了大小月的判斷，讓使用者可以避免更多的失誤。
- heightCheck(): 這裡確認了身高是否在一定的範圍內，並且提醒使用者要使用cm當作單位。
- weightCheck(): 用以確認體重不會出現不合理的值，以及提醒使用kg。

- methods of Persons

```java
private double calculateBMI() {
    return this.weight / ((this.height/100) * (this.height/100));
}

public String getName() {
    return this.name;
}

public int getAge() {
    return this.age;
}

public double getBMI() {
    return this.bmi;
}

public String getBirthDate() {
    return this.monthOfBirth + "/" + this.dayOfBirth + "/" + this.yearOfBirth;
}

public String showMyInfo() {
    DecimalFormat df = new DecimalFormat("0.00");
    String correctBmi = df.format(this.bmi);
    String correctHeight = df.format(this.height);
    String correctWeight = df.format(this.weight);

    return "\nName: " + this.name + "\n" +
            "Age: " + this.age + "\n" +
            "Height: " + correctHeight + " cm\n" +
            "Weight: " + correctWeight + " kg\n" +
            "BMI: " + correctBmi + "\n" +
            "Birth date: " + this.monthOfBirth + "/" + this.dayOfBirth + "/" + this.yearOfBirth;
}
```

- calculateBMI(): 計算使用者的BMI。
- getName(), getAge(), getBMI(): 為getter，可以分別取的各自的屬性資訊。
- getBirthDate(): 輸出使用者的生日資訊。
- showMyInfo(): 可以將使用者的資訊一併輸出。

Person - 程式執行結果

- 正常輸入

```
/Library/Java/JavaVirtualMachines/adoptopenjdk-16.jdk/Contents/Home/bin/java
Enter name: alec wu
Enter year of birth: 2003
Enter month of birth: 1
Enter day of birth: 21
Enter height(cm): 169
Enter weight(kg): 60

Name: alec wu
Age: 20
Height: 169.00 cm
Weight: 60.00 kg
BMI: 21.01
Birth date: 1/21/2003
```

- 發生錯誤輸入

```
/Library/Java/JavaVirtualMachines/adoptopenjdk-16.jdk/Contents/Home/bin/java
Enter name:
Invalid input
Enter name: alec wu
Enter year of birth: 2025
Year of birth cannot be less than 1850 or greater than current year 2023
Enter year of birth: 2003
Enter month of birth: 13
Month of birth cannot be less than 1 or greater than 12
Enter month of birth: 12
Enter day of birth: 20
Enter height(cm): 169
Enter weight(kg): 50

Name: alec wu
Age: 20
Height: 169.00 cm
Weight: 50.00 kg
BMI: 17.51
Birth date: 12/20/2003
```

Person - Unit Test

- 測試報告書

PersonTest: 13 total, 13 passed 97 ms

Collapse | Expand

/Library/Java/JavaVirtualMachines/adoptopenjdk-16.jdk/Contents/Home/bin/java -ea -Didea.test.cyclic.buffer.size=1048576 -javaagent:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar=61273:/Applications/IntelliJ IDEA CE.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/alecwu/.m2/repository/org/junit/platform/junit-platform-launcher/1.9.1/junit-platform-launcher-1.9.1.jar:/Applications/IntelliJ IDEA CE.app/Contents/lib/idea_rt.jar:/Applications/IntelliJ IDEA CE.app/Contents/plugins/junit/lib/junit5-rt.jar:/Applications/IntelliJ IDEA CE.app/Contents/plugins/junit/lib/junit-rt.jar:/Users/alecwu/Desktop/軟體測試/HW01/target/test-classes:/Users/alecwu/Desktop/軟體測試/HW01/target/classes:/Users/alecwu/.m2/repository/org/junit/jupiter/junit-jupiter-engine/5.9.1/junit-jupiter-engine-5.9.1.jar:/Users/alecwu/.m2/repository/org/junit/platform/junit-platform-engine/1.9.1/junit-platform-engine-1.9.1.jar:/Users/alecwu/.m2/repository/org/opentest4j/opentest4j/1.2.0/opentest4j-1.2.0.jar:/Users/alecwu/.m2/repository/org/junit/platform/junit-platform-commons/1.9.1/junit-platform-commons-1.9.1.jar:/Users/alecwu/.m2/repository/org/junit/jupiter/junit-jupiter-api/5.9.1/junit-jupiter-api-5.9.1.jar:/Users/alecwu/.m2/repository/org/apiguardian/apiguardian-api/1.1.2/apiguardian-api-1.1.2.jar com.intellij.rt.junit.JUnitStarter -ideVersion5 -junit5 org.example.PersonTest Process finished with exit code 0

monthCheck()	passed	46 ms
getAge()	passed	2 ms
getBMI()	passed	3 ms
getBirthDate()	passed	10 ms
heightCheck()	passed	2 ms
yearCheck()	passed	4 ms
getName()	passed	1 ms
main()	passed	1 ms
showMyInfo()	passed	18 ms
isLeapYear()	passed	3 ms
nameCheck()	passed	2 ms
dayCheck()	passed	3 ms
weightCheck()	passed	2 ms

Generated by IntelliJ IDEA on 2023/10/14 上午5:38

這裡進行了13種測試，而測試結果為全部通過，下個階段將介紹每個測試的詳細過程與設計。

- 測試前初始化

```
    Person person = new Person();
    Person personA = new Person();
    Person personB = new Person();
    Person personC = new Person();
    Person personD = new Person();
    Person personE = new Person();

    @BeforeEach
        // To initialize the test
    void init() {
        person = new Person("Nguyen Van A", 2000, 12, 31, 170, 60);
        personA = new Person("Alec Wu", 2023, 1, 21, 170, 60);
        personB = new Person("", 2000, 12, 31, 170, 60);
        personC = new Person("Emily Chen", 1991, 5, 32, 169, 50);
        personD = new Person("Elle CX", 1993, 2, 29, 168, 50);
        personE = new Person("Ashley CA", 1992, 2, 29, 168, 50);

//        System.out.println("init");
        System.out.println("[>] Init MainTest\n");
    }
```

這裡使用了多個物件，為每個人設都進行測試，也同樣地在測試前讓物件保持乾淨，並提示使用者。

- getter測試

getAge() passed 2 ms

```
[>] Init MainTest
[+] Here is getAge test
This test initially bypasses judgment and solely assesses whether someone can retrieve the age
[v] getAge test passed
```

getBMI() passed 3 ms

```
[>] Init MainTest
[+] Here is getBMI test
This test initially bypasses judgment and solely assesses whether someone can retrieve the BMI
[v] getBMI test passed
```

getBirthDate() passed 10 ms

```
[>] Init MainTest
[+] Here is getBirthDate test
This test initially bypasses judgment and solely assesses whether someone can retrieve the birth date
[v] getBirthDate test passed
```

getName() passed 1 ms

```
[>] Init MainTest
[+] Here is getName test
This test initially bypasses judgment and solely assesses whether someone can retrieve the name
[v] getName test passed
```

由測試報告可以得知所以的輸出結果，而在裡面時我們分別針對多個人設做測試，並且確定都可以取得到所要的資料。

```java
@Test
void getName() {
    System.out.println("[+] Here is getName test");
    System.out.println("This test initially bypasses judgment and solely assesses whether someone can retrieve the name");
    assertEquals("Nguyen Van A", person.getName());
    assertEquals("Alec Wu", personA.getName());
    assertEquals("", personB.getName());
    assertEquals("Emily Chen", personC.getName());
    assertEquals("Elle CX", personD.getName());
    assertEquals("Ashley CA", personE.getName());

    System.out.println("[v] getName test passed");
}

@Test
void getAge() {
    System.out.println("[+] Here is getAge test");
    System.out.println("This test initially bypasses judgment and solely assesses whether someone can retrieve the age");

    LocalDate currentDate = LocalDate.now();
    assertEquals(currentDate.getYear() - 2000, person.getAge());
    assertEquals(currentDate.getYear() - 2023, personA.getAge());
    assertEquals(currentDate.getYear() - 2000, personB.getAge());
    assertEquals(currentDate.getYear() - 1991, personC.getAge());
    assertEquals(currentDate.getYear() - 1993, personD.getAge());
    assertEquals(currentDate.getYear() - 1992, personE.getAge());

    System.out.println("[v] getAge test passed");
}

@Test
void getBMI() {
    System.out.println("[+] Here is getBMI test");
    System.out.println("This test initially bypasses judgment and solely assesses whether someone can retrieve the BMI");

    assertEquals(20.761245674740486, person.getBMI());
    assertEquals(20.761245674740486, personA.getBMI());
    assertEquals(20.761245674740486, personB.getBMI());
    assertEquals(17.506389832288786, personC.getBMI());
    assertEquals(17.71541950113379, personD.getBMI());
    assertEquals(17.71541950113379, personE.getBMI());

    System.out.println("[v] getBMI test passed");
}

@Test
void getBirthDate() {
    System.out.println("[+] Here is getBirthDate test");
    System.out.println("This test initially bypasses judgment and solely assesses whether someone can retrieve the birth date");

    assertEquals("12/31/2000", person.getBirthDate());
    assertEquals("1/21/2023", personA.getBirthDate());
    assertEquals("12/31/2000", personB.getBirthDate());
    assertEquals("5/32/1991", personC.getBirthDate());
    assertEquals("2/29/1993", personD.getBirthDate());
    assertEquals("2/29/1992", personE.getBirthDate());

    System.out.println("[v] getBirthDate test passed");
}
```

- checker測試

monthCheck() passed 46 ms

```
[>] Init MainTest
[+] Here is monthCheck test
To testing the monthCheck function
Case #1: pass
Input is 0, Exception message: Month of birth cannot be less than 1 or greater than 12
Case #2: pass
Input is 13, Exception message: Month of birth cannot be less than 1 or greater than 12
[v] monthCheck test passed
```

heightCheck() passed 2 ms

```
[>] Init MainTest
[+] Here is heightCheck test
To testing the heightCheck function
Case #1: pass
Input is 0, Exception message: Height cannot be less than 0 or greater than 300
Please use cm
Case #2: pass
Input is 1.68, Exception message: Height cannot be less than 0 or greater than 300
Please use cm
[v] heightCheck test passed
```

yearCheck() passed 4 ms

```
[>] Init MainTest
[+] Here is yearCheck test
To testing the yearCheck function
Case #1: pass
Input is 1849, Exception message: Year of birth cannot be less than 1850 or greater than current year 2023
Case #2: pass
Input is 2027, Exception message: Year of birth cannot be less than 1850 or greater than current year 2023
[v] yearCheck test passed
```

isLeapYear() passed 3 ms

```
[>] Init MainTest
[+] Here is isLeapYear test
To testing the isLeapYear function
[v] isLeapYear test passed
```

nameCheck() passed 2 ms

```
[>] Init MainTest
[+] Here is nameCheck test
To testing the nameCheck function
Case #1: pass
Input is empty, Exception message: Name cannot be less than 1 or greater than 100 characters
[v] nameCheck test passed
```

dayCheck() passed 3 ms

```
[>] Init MainTest
[+] Here is dayCheck test
To testing the dayCheck function
Case #1: pass
Date: 12/31/2000
Input is 0, Exception message: Day of birth cannot be less than 1 or greater than 31
Case #2: pass
Date: 12/31/2000
Input is 32, Exception message: Day of birth cannot be less than 1 or greater than 31
Case #3: pass
Date: 12/31/2000
Input is 29, Exception message: Day of birth cannot be greater than 28
Case #4: pass
Date: 12/31/2000
Input is 30, Exception message: Day of birth cannot be greater than 29
[v] dayCheck test passed
```

weightCheck() passed 2 ms

```
[>] Init MainTest
[+] Here is weightCheck test
To testing the weightCheck function
Case #1: pass
Input is 0, Exception message: Weight cannot be less than 0 or greater than 500
Please use kg
Case #2: pass
Input is 900, Exception message: Weight cannot be less than 0 or greater than 500
Please use kg
[v] weightCheck test passed
```

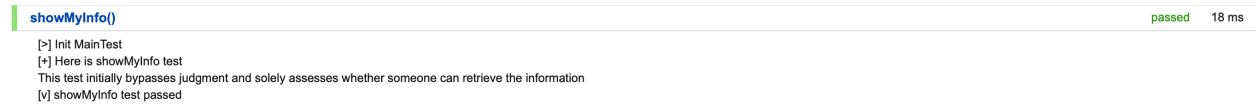
這裡採用許多的極端值以測試exception是否可以正常運作，而程式設計以及所採用的值，如下頁的片段程式碼。

```java
@Test
void nameCheck() {
    System.out.println("[+] Here is nameCheck test");
    System.out.println("To testing the nameCheck function");

    Exception exception = assertThrows(Exception.class, () -> {
        person.nameCheck("");
    });
    System.out.println("Case #1: pass");
    System.out.println("Input is empty, " + "Exception message: " + exception.getMessage());

    System.out.println("[v] nameCheck test passed");
}

@Test
void yearCheck() {
    System.out.println("[+] Here is yearCheck test");
    System.out.println("To testing the yearCheck function");

    Exception exception = assertThrows(Exception.class, () -> {
        person.yearCheck(1849);
    });
    System.out.println("Case #1: pass");
    System.out.println("Input is 1849, " + "Exception message: " + exception.getMessage());

    exception = assertThrows(Exception.class, () -> {
        person.yearCheck(2027);
    });
    System.out.println("Case #2: pass");
    System.out.println("Input is 2027, " + "Exception message: " + exception.getMessage());

    System.out.println("[v] yearCheck test passed");
}
```

```java
@Test
void monthCheck() {
    System.out.println("[+] Here is monthCheck test");
    System.out.println("To testing the monthCheck function");

    Exception exception = assertThrows(Exception.class, () -> {
        person.monthCheck(0, 2000);
    });
    System.out.println("Case #1: pass");
    System.out.println("Input is 0, " + "Exception message: " + exception.getMessage());

    exception = assertThrows(Exception.class, () -> {
        person.monthCheck(13, 2000);
    });
    System.out.println("Case #2: pass");
    System.out.println("Input is 13, " + "Exception message: " + exception.getMessage());

    System.out.println("[v] monthCheck test passed");
}

@Test
void isLeapYear() {
    System.out.println("[+] Here is isLeapYear test");
    System.out.println("To testing the isLeapYear function");

    assertEquals(false, person.isLeapYear(2001));
    assertEquals(true, person.isLeapYear(2000));
    assertEquals(false, person.isLeapYear(1900));
    assertEquals(true, person.isLeapYear(2004));
    assertEquals(false, person.isLeapYear(2005));

    assertEquals(false, personD.isLeapYear(personD.yearOfBirth));
    assertEquals(true, personE.isLeapYear(personE.yearOfBirth));

    System.out.println("[v] isLeapYear test passed");
}
```

```java
@Test
void dayCheck() {
    System.out.println("[+] Here is dayCheck test");
    System.out.println("To testing the dayCheck function");

    Exception exception = assertThrows(Exception.class, () -> {
        person.dayCheck(1, 0, 2000);
    });
    System.out.println("Case #1: pass");
    System.out.println("Date: " + person.getBirthDate());
    System.out.println("Input is 0, " + "Exception message: " + exception.getMessage());

    exception = assertThrows(Exception.class, () -> {
        person.dayCheck(1, 32, 2000);
    });
    System.out.println("Case #2: pass");
    System.out.println("Date: " + person.getBirthDate());
    System.out.println("Input is 32, " + "Exception message: " + exception.getMessage());

    exception = assertThrows(Exception.class, () -> {
        person.dayCheck(2, 29, 2001);
    });
    System.out.println("Case #3: pass");
    System.out.println("Date: " + person.getBirthDate());
    System.out.println("Input is 29, " + "Exception message: " + exception.getMessage());

    exception = assertThrows(Exception.class, () -> {
        person.dayCheck(2, 30, 2000);
    });
    System.out.println("Case #4: pass");
    System.out.println("Date: " + person.getBirthDate());
    System.out.println("Input is 30, " + "Exception message: " + exception.getMessage());

    System.out.println("[v] dayCheck test passed");
}

@Test
void heightCheck() {
    System.out.println("[+] Here is heightCheck test");
    System.out.println("To testing the heightCheck function");

    Exception exception = assertThrows(Exception.class, () -> {
        person.heightCheck(0);
    });
    System.out.println("Case #1: pass");
    System.out.println("Input is 0, " + "Exception message: " + exception.getMessage());

    exception = assertThrows(Exception.class, () -> {
        person.heightCheck(1.68);
    });
    System.out.println("Case #2: pass");
    System.out.println("Input is 1.68, " + "Exception message: " + exception.getMessage());

    System.out.println("[v] heightCheck test passed");
}

@Test
void weightCheck() {
    System.out.println("[+] Here is weightCheck test");
    System.out.println("To testing the weightCheck function");

    Exception exception = assertThrows(Exception.class, () -> {
        person.weightCheck(0);
    });
    System.out.println("Case #1: pass");
    System.out.println("Input is 0, " + "Exception message: " + exception.getMessage());

    exception = assertThrows(Exception.class, () -> {
        person.weightCheck(900);
    });
    System.out.println("Case #2: pass");
    System.out.println("Input is 900, " + "Exception message: " + exception.getMessage());

    System.out.println("[v] weightCheck test passed");
}
```

- showMyInfo()



這裡一樣使用多筆資料來確認是否所輸出的內容為一致。

```java
@Test
 void showMyInfo() {
     System.out.println("[+] Here is showMyInfo test");
     System.out.println("This test initially bypasses judgment and solely assesses whether someone can retrieve the information");

     assertEquals("\nName: Nguyen Van A\n" +
             "Age: 23\n" +
             "Height: 170.00 cm\n" +
             "Weight: 60.00 kg\n" +
             "BMI: 20.76\n" +
             "Birth date: 12/31/2000", person.showMyInfo());
     assertEquals("\nName: Alec Wu\n" +
             "Age: 0\n" +
             "Height: 170.00 cm\n" +
             "Weight: 60.00 kg\n" +
             "BMI: 20.76\n" +
             "Birth date: 1/21/2023", personA.showMyInfo());
     assertEquals("\nName: \n" +
             "Age: 23\n" +
             "Height: 170.00 cm\n" +
             "Weight: 60.00 kg\n" +
             "BMI: 20.76\n" +
             "Birth date: 12/31/2000", personB.showMyInfo());
     assertEquals("\nName: Emily Chen\n" +
             "Age: 32\n" +
             "Height: 169.00 cm\n" +
             "Weight: 50.00 kg\n" +
             "BMI: 17.51\n" +
             "Birth date: 5/32/1991", personC.showMyInfo());
     assertEquals("\nName: Elle CX\n" +
             "Age: 30\n" +
             "Height: 168.00 cm\n" +
             "Weight: 50.00 kg\n" +
             "BMI: 17.72\n" +
             "Birth date: 2/29/1993", personD.showMyInfo());
     assertEquals("\nName: Ashley CA\n" +
             "Age: 31\n" +
             "Height: 168.00 cm\n" +
             "Weight: 50.00 kg\n" +
             "BMI: 17.72\n" +
             "Birth date: 2/29/1992", personE.showMyInfo());

     System.out.println("[v] showMyInfo test passed");
 }
```

最後，每筆測試都如期通過，也代表程式在未來可以正常運作。

心得

經由這次的軟體測試作業一，讓我明白何謂軟體測試，也體會到一個好的開發者，不僅要在寫程式考慮效能、可以work，也要為自己的程式可以確保是正確的，其中一開始在寫測資的時候有點難想，最後根據以前演算法競賽的想法，來檢驗自己的測試法是否可以運作正常。

在寫測試的時候，也讓我想到在公司工作時，幫團隊用Jenkins設立了自動化測試及部署，也讓我開始思考如何將之前的經驗用在這次的作業上，因為在開發時，自動化的工具可以讓開發者也減少許多的經歷與時間，來增加工作的效率。

最後，這次的作業使我受益良多，不僅讓我更加明白如何寫好一個測試，讓我之後在工作上可以運用所學，確保團隊所開發的程式碼，可以在一定的品質，另外，也讓我去嘗試結合自動化的工具，來增加效率，讓我在開發上也有許多的進步，也期待老師於下次可以教授我們更多有關測試以及自動化的工具。