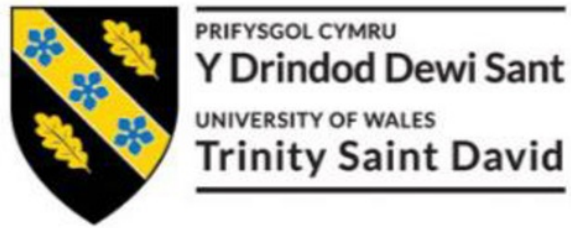


James Jones Programming Assignment

by James Robert Tudor Jones

FILE	11685_JAMES_ROBERT_TUDOR_JONES_JAMES_JONES_PROGRAMMIN G_ASSIGNMENT_159704_1266753519.DOCX		
TIME SUBMITTED	11-JAN-2019 09:50AM (UTC+0000)	WORD COUNT	4986
SUBMISSION ID	98777512	CHARACTER COUNT	31460



University of Wales Trinity Saint David Swansea

MSc Software Engineering Principles

Programming and Software Testing

Student: James Jones

ID: P138563

Contents

Contents.....	1
1.0 Introduction	3
Part 1.....	3
2.0 DVLA Testing processes & techniques	3
2.1 Static Testing.....	4
2.2 Unit testing.....	4
2.3 User interface testing.....	5
2.4 Acceptance testing.....	6
2.5 Integration testing	6
2.6 Load Testing	7
2.7 Stress testing.....	7
2.8 Soak Testing	8
3.0 Conclusion.....	8
Part 2.....	9
1.0 Software development	9
1.1 High cohesion.....	9
1.11 Functional Cohesion.....	10
1.12 Sequential Cohesion	10
1.13 Communicational Cohesion	10
1.14 Procedural Cohesion	10
1.15 Temporal Cohesion	10
1.16 Logical Cohesion.....	10
1.17 Coincidental Cohesion	10
1.2 Loose Coupling.....	10
1.3 Data Hiding.....	12
1.31 Data hiding vs Encapsulation	12
1.4 Information hiding	13

2.0 Conclusion.....	13
3.0 Reference List.....	13
4.0 Appendices.....	16

1.0 Introduction

This report will be split into two Parts, part 1 will look at Testing processes and techniques which are undertaken by the DVLA. The Testing process will give detail the different type of software development tests that are used throughout the lifecycle of the systems used in the DVLA. Part 2 will look at software development with the use of High Cohesion, loose coupling, data hiding and information hiding and conclude on why these topics are very important with the software development cycle.

Part 1

2.0 DVLA Testing processes & techniques

There are many testing processes and techniques used within the DVLA which include static testing, unit testing, user interface testing, acceptance testing, integration testing, load testing, stress testing and soak testing.

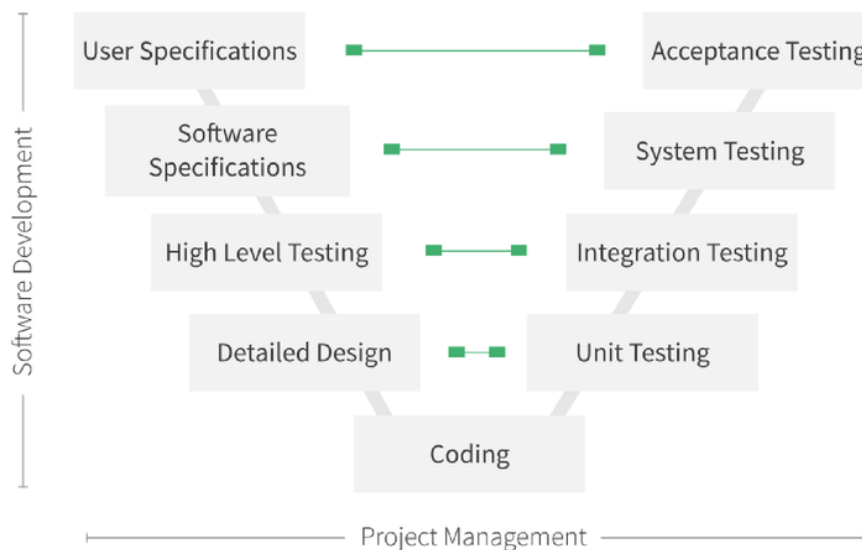


Figure 1: Testing lifecycle

[1]

Figure 1 gives an overview of how the testing can be arranged and represents the lifecycle, these tests can be looked at with more detail.

2.1 Static Testing

Static testing assists a user to fix their own faults early in the development, the DVLA use this type of testing at the verification level. Tools QA explains 'Static Testing Techniques provide a powerful way to improve the quality and productivity of software development by assisting engineers to recognize and fix their own defects early in the software development process.' [2] There are many software packages that can be used for static testing the DVLA use many. One of them being Apache Jmeter. Static testing is useful due to the fact it is used early in the life cycle, this establishes quality issues early enough to eliminate problems later on in development. Rework cost is lower when static testing is taken out. techniques used with static testing involve a step through and certain processes.

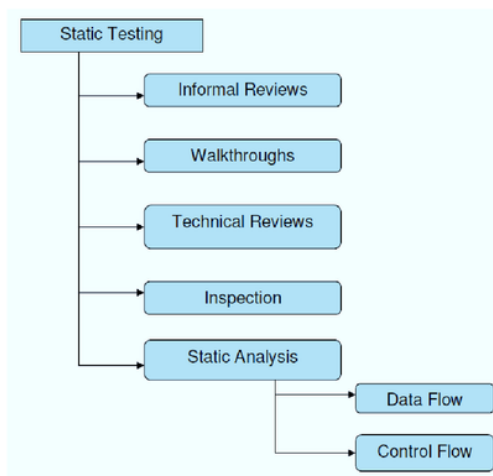


Figure 2: Static Testing Techniques

[3]

Figure 2 represents the technique used within static testing there can be different representations of the testing but generally this is the baseline that static testing follows.

2.2 Unit testing

Unit testing is another type of test which is used within the DVLA. The unit test is effectively used to detect error of every function. This method of testing can be broken down and looked at with the use of static testing and dynamic testing in one. [4] the purpose of unit testing is to validate each unit of the software. A unit test can be very small sometimes only testing something with the fewest inputs with a single output. A typical unit test has three stages which can be demonstrated in a block diagram: -

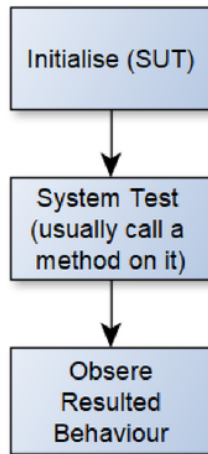


Figure 3: Basic unit test technique

Figure 3 from top to bottom the first stage is to Initialise the test this is formerly known as SUT¹ system under test. The SUT which can also be defined as application under test AUT² method under test MUT and class under test CUT is usually a user design to define the perspective of the test, the SUT represents all of the actors. The second stage is the system test this part is usually calling a method onto the code to make a test against it for example having input integers and create a math test for example then lastly observing the resulted behaviour to recognise a pass or fail. The types of software used at the DVLA include Junit and Selenium.

2.3 User interface testing

User face testing within the DVLA is implemented by using a GUI to test the coding. At this stage the main focus is to detect visual flaws, any structural defects and also to identify if the UI is working correctly with navigation and full app functionality. There are several approaches that can be used for example how the UI responds to an action being made such as a mouse click, keyboard press or touch gestures. Xbsoftware states 'We test whether a program behaves as expected and whether interface elements are displayed on different devices properly when a user performs some actions.'^[5]. The main software packages used within the DVLA include Cucumber and selenium.³

2.4 Acceptance testing

Acceptance testing is to check that the system is tested for acceptability. The test is to evaluate the systems compliance with the business requirements and to determine if it can be accepted for delivery.[6] Acceptance testing usually uses the black box testing methodology. Black box testing is essentially a software method test which is being tested but is not known by the tester.

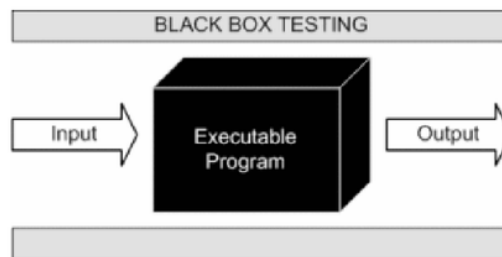


Figure 4: Black box Test

[7]

From a tester point of view, the software program cannot be seen hence the name black box. The errors that this type of test include: -

- Incorrect or missing functions.
- Interface error.
- Errors in data structures or external database access.
- Behaviour or performance errors.
- Initialization and termination errors.

The acceptance testing is demanded by the customer. The test gives an outcome to gain confidence in the product it ensures that the product is working in the way it has to.

2.5 Integration testing

Integration testing is used to combine individual units and test them as a group. This type of testing is used in the DVLA and exposes any faults or errors when integrated together. Integration testing can be created with the use of stub tests. This type of testing makes sure that components work properly. This test can be started soon as the modules to be tested are available. It doesn't require other modules to be completed this test can also detect errors related to the interface. Software testing help explain 'Test Integration is an important part of the testing cycle as it makes it easier to find the defect when two or more modules are integrated in order to integrate all the modules all together in the first step itself.' [8]

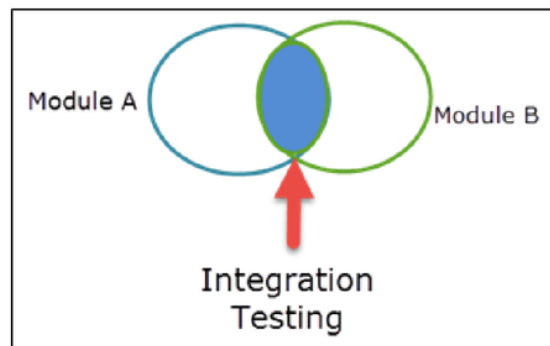


Figure 5: Integration Test representation

[9]

2.6 Load Testing

Load testing is used to mimic the real world. A big load is used for example to mimic hundreds of users doing something on one thing. It accurately measures the capability of a single system. It can give a result to see how well the particular system will function in the real world. Microsoft states 'The load test verifies application behaviour under normal and peak load conditions'. [10] There are many examples of different types of load testing and is also formerly known as a performance testing. Different examples include: -

- Flooding lots of inputs on a system
- Load testing is conducted to verify that your application can meet your desired performance objectives
- Endurance testing may be used to calculate Mean Time Between Failure (MTBF), Mean Time to Failure (MTTF), and similar metrics.

2.7 Stress testing

Load view testing explains 'A stress test is a test designed to increase the number of simultaneous requests on a system past a point where performance is degraded, possibly even to the point of complete failure.' [11] this type of test can really test the longevity of the system which is a very important step before delivery of any system. Pushing a system beyond normal of peak load conditions can result in different behaviours which need to be addressed, it can reveal application bugs Microsoft describes 'These bugs can include such things as synchronization issues, race

conditions, and memory leaks. Stress testing enables you to identify your application's weak points, and shows how the application behaves under extreme load conditions.'[10]

2.8 Soak Testing

Soak testing is like a form of stress testing but this time it is used over a long period of time to see how a system behaves. Techtarget explains 'The process informs software testers whether an application performs satisfactorily after hours -- or days -- of usage. Soak testing uncovers software issues that only become apparent after it runs for extended periods of time, such as a gradual memory leak.'[12] the DVLA have this test in place but it was mentioned that this test can come at a very high cost if run for days therefore financially this test is chosen wisely and given time constraints.

3.0 Conclusion

There are many different test methodologies that can be used in any organisation the DVLA uses many different techniques. Testing must be taken out to reduce the possibility of the end product falling over and crating errors and inconvenience for an end user or customer. Interestingly while looking at different types of tests, the technique of creating a test first before any code has been created has major advantages as well as testing the code afterwards.

Part 2

1.0 Software development

There are different styles when undertaking software development among those styles there are the subject of high cohesion, loose coupling, data hiding and information hiding which all serve an important purpose within software engineering these topics can be looked at a lot closer to understand why they are used.

1.1 High cohesion

High cohesion is the preferred software development compared to low cohesion. A Low cohesion would mean a class or function which has a great variety of actions being too broad, whereas a high cohesion means the class is focused on its desired objective. A good software design will have high Josikakar describes 'Basically, cohesion is the internal glue that keeps the module together. A good software design will have high cohesion.' [13]

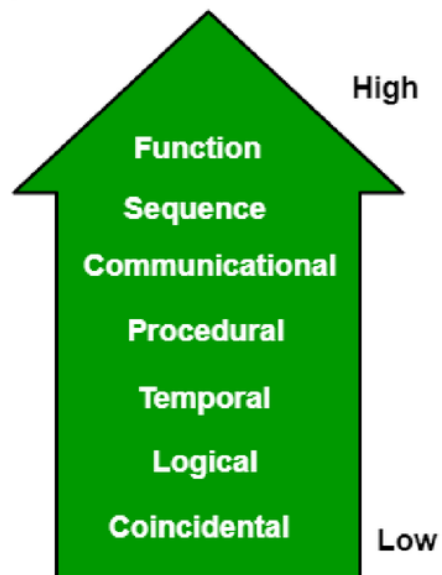


Figure 6: Cohesion stages [13]

Figure 6 demonstrates different levels of cohesion from Low to high cohesion. Each stage can be looked into with detail.

1.11 Functional Cohesion

All elements for a single class is contained in the component. Functional cohesion performs the task and functions. This cohesion is the ideal and.

1.12 Sequential Cohesion

Some data will become the input for another part. For example, there will be data flow between the parts.

1.13 Communicational Cohesion

A pair of parts will operate with the same data input or share the same output data.

1.14 Procedural Cohesion

Ensures the order of result, the actions are connected weakly and will be unlikely reusable.

1.15 Temporal Cohesion

Related by the timing involved, all the tasks are needed to be in the same time-span. Different activities occur many times.

1.16 Logical Cohesion

Logically related but not functional, the operations are related but the functions are different.

1.17 Coincidental Cohesion

There is no relationship between the elements, this is the worst for of cohesion.

The main use of cohesion is all about the internal consistency of a class or method

1.2 Loose Coupling

Interconnection components in a system or a network which can also be formerly known as elements will depend on each other. Coupling has a link of direct knowledge that one element has of another. Techtarget explains 'The goal of a loose coupling architecture is to reduce the risk that a change made within one element will create unanticipated changes within other elements.' [14] while limiting these interconnections with the use of loose coupling, the use of low coupling also simplifies tests. Alho Pekka, Mattila Jouni argue 'Another benefit of loose coupling between services is capability to have multiple copies running on more than one computer at once, providing redundant processing or hot backup capabilities.' [15] any piece of coding has some sort of coupling but whether it the best type of coupling is the question. The right kind of coupling is what is wanted

to have a good piece of code. A piece of coding that has no coupling in a system is a system that does nothing.[16]

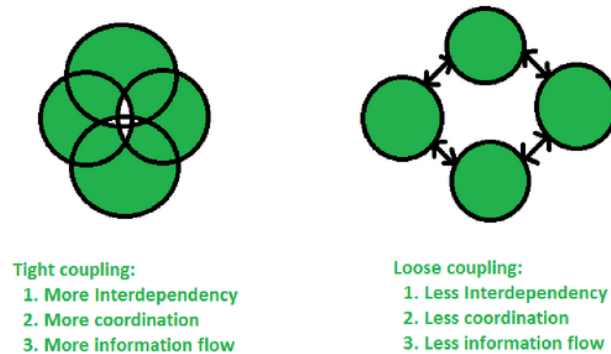


Figure 7: Tight and Loose coupling

[17]

Figure 7 investigates tight coupling compared to loose coupling when inspecting a tight coupled system there is need of more dependencies more coordination and more information flow which is bad for coding meaning that if tight coupling is used the classes and methods can depend on each other to much lacking flexibility and can cause further errors if something was say to be changed it can affect all the code. With the use of loose coupling there is less dependencies, less coordination and less information flow this way of coding creates flexibility, it can reduce the errors because if something was to be changed there will be less classes or methods in relation to the change meaning smaller changes. Loose coupling keeps the relationship entities clear and easier to work with in the long run for a coder.[16]

1.3 Data Hiding

Data hiding was introduced as part of the Object oriented programming (OOP) methodology, the main purpose of data hiding is a way of hiding data to the outside world, this allows protection of data. When a coder makes a piece of code public this can be easily accessed and changed at any time. When using the method of data hiding the piece of code is protected and restricts outside entities from making any modifications.[18]

1.31 Data hiding vs Encapsulation

For the purpose of comparing data hiding and encapsulation the following code is written in C++.

The code gives clear differences between the two instances.

encapsulation

```
1. class MyClass {
2. public: // members functions and stuff here
3. private: int foo; // members are visible (not hidden)
4. int bar; // but are still private (encapsulated)
5. };
```

Data hiding

```
1. class MyClass {
2. public: // member functions and stuff here private: struct Data;
3. Data* data; // user cannot see what 'data' this class uses // therefore
   it's hidden
4. };
5. //===== // then, in the .cpp file... the struct would need to be
   defined
6. struct MyClass::Data { int foo; int bar; }; [19]
```

with the encapsulation method the variable foo has still been assigned as private but this piece of data is still visible and not hidden from the outside world whereas when looking at the data hiding method the Data* is hidden and cannot be seen by the outside world giving a level of protection.

1.4 Information hiding

Information hiding can easily be mistaken as data hiding, the main use of information hiding is to reduce complexities and can make a function easier to use. Encapsulation is a known technique when using information hiding. Itkonen, Juha Mäntylä, Mika details 'Information hiding is a very powerful and flexible technique that can be employed in various ways to protect valuable information in different areas such as copyright protection, secure communication, and authentication.' [20]

2.0 Conclusion

The ideal coder will want to aim for High cohesion and loose coupling integrated together for the best useable system. This will give a strong code output making sure that high cohesion gives the required relation within the classes and loose coupling to make sure there is not too many relations between all classes reducing errors and making the code easier to navigate. Data hiding and information hiding is especially important with many instances with code. The use of data hiding is important to hide data from particular users for data protection and information hiding reduces complexity of code.

3.0 Reference List

- [1] J. Itkonen and M. V. Mäntylä, "Are test cases needed? Replicated comparison between exploratory and test-case-based software testing," *Empir. Softw. Eng.*, vol. 19, no. 2, pp. 303–342, Apr. 2014.
- [2] "What is Static Testing?" [Online]. Available: <https://www.toolsqa.com/software-testing/static-testing/>. [Accessed: 02-Jan-2019].
- [3] "What is Static Testing? What is a Testing Review?" [Online]. Available: <https://www.guru99.com/testing-review.html>. [Accessed: 02-Jan-2019].
- [4] Y. Li and D. Liu, "The Application of Testbed Tbrun in Unit Test of Aviation Software," *Appl. Mech. Mater.*, vol. 347–350, 2013.
- [5] "User Interface (UI) Testing - XB Software." [Online]. Available: <https://xbsoftware.com/qa-software-testing/full-qa-cycle/user-interface-testing/>. [Accessed: 02-Jan-2019].
- [6] "Acceptance Testing - Software Testing Fundamentals." [Online]. Available: <http://softwaretestingfundamentals.com/acceptance-testing/>. [Accessed: 02-Jan-2019].

- [7] "Black Box Testing - Software Testing Fundamentals." [Online]. Available: <http://softwaretestingfundamentals.com/black-box-testing/>. [Accessed: 02-Jan-2019].
- [8] "What is Integration Testing (Tutorial with Integration Testing Example)." [Online]. Available: <https://www.softwaretestinghelp.com/what-is-integration-testing/>. [Accessed: 02-Jan-2019].
- [9] "What is System Integration Testing (SIT) with Example." [Online]. Available: <https://www.guru99.com/system-integration-testing.html>. [Accessed: 02-Jan-2019].
- [10] "Chapter 2 – Types of Performance Testing | Microsoft Docs." [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions/msp-n-p/bb924357\(v=pandp.10\)](https://docs.microsoft.com/en-us/previous-versions/msp-n-p/bb924357(v=pandp.10)). [Accessed: 02-Jan-2019].
- [11] "Load Testing vs Stress Testing." [Online]. Available: <https://www.loadview-testing.com/load-testing-vs-stress-testing/>. [Accessed: 02-Jan-2019].
- [12] "What is soak testing? - Definition from WhatIs.com." [Online]. Available: <https://searchsoftwarequality.techtarget.com/definition/Soak-testing>. [Accessed: 02-Jan-2019].
- [13] Josikakar, "Software Engineering | Coupling and Cohesion - GeeksforGeeks." [Online]. Available: <https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/>. [Accessed: 02-Jan-2019].
- [14] "What is loose coupling? - Definition from WhatIs.com." [Online]. Available: <https://searchnetworking.techtarget.com/definition/loose-coupling>. [Accessed: 03-Jan-2019].
- [15] P. Alho and J. Mattila, "Software fault detection and recovery in critical real-time systems: An approach based on loose coupling," *Fusion Eng. Des.*, vol. 89, no. 9–10, pp. 2272–2277, 2014.
- [16] "Quality Code is Loosely Coupled." [Online]. Available: <https://tobeagile.com/2017/05/10/quality-code-is-loosely-coupled/>. [Accessed: 04-Jan-2019].
- [17] "Coupling in Java - GeeksforGeeks." [Online]. Available: <https://www.geeksforgeeks.org/coupling-in-java/>. [Accessed: 04-Jan-2019].
- [18] "oop - Encapsulation vs Data Hiding - Java - Stack Overflow." [Online]. Available: <https://stackoverflow.com/questions/12013448/encapsulation-vs-data-hiding-java>. [Accessed: 10-Jan-2019].

- [19] "What are some clear differences between data hiding, data encapsulation and data abstraction? - Quora." [Online]. Available: <https://www.quora.com/What-are-some-clear-differences-between-data-hiding-data-encapsulation-and-data-abstraction>. [Accessed: 10-Jan-2019].
- [20] M. Taleby Ahvanooley, Q. Li, H. J. Shim, and Y. Huang, "A Comparative Analysis of Information Hiding Techniques for Copyright Protection of Text Documents," *Secur. Commun. Networks*, vol. 2018, pp. 1–22, Apr. 2018.

4.0 Appendices

4.1 Appendix 1 Part 1 Coding

```
package MSocSoftwareDevelopment;
import java.util.*;

public class Main
{
    private static Scanner stdin = new Scanner(System.in);

    /**
     * Main program that calls the functions and passes parameters.
     */
    public static void main(String[] args)
    {
        int count=0 , countA=0 ,countB = 0;
        int amountA=0, amountB=0;
        int setAAmount=0, setBAmount=0;
        int chosenMenu;
        int[] setA = new int[0];
        int[] setB = new int[0];
        boolean check;

        setAAmount = EnterAAmount(amountA);
        setBAmount = EnterBAmount(amountB);
        setA = SetACount(count,setAAmount,setA);
        setB = SetBCount(count,setBAmount,setB);
        PerformFuntion(chosenMenu = 0,count,countA, countB ,setAAmount,setBAmount,setA,setB, check = false);
    }

    /**
     * User chooses which option to use from list
     * returns integer chosen menu
     */
    static int ChooseFunction()
    {
        int chosenMenu = 0;

        System.out.println();
        System.out.println("*****MENU*****");
        System.out.println("1. Display the numbers in both sets");
        System.out.println("2. Display maximum and minimum values");
        System.out.println("3. Display Average of the numbers in each set");
        System.out.println("4. Display if Set A and Set B overlap");
        System.out.println("5. Display intersection of A and B if possible");
        System.out.println("6. Display the set difference of A and B if possible");
        System.out.println("7. Quit");
        System.out.println("Please Select option");

        try
        {
            chosenMenu = stdin.nextInt();
        }
        catch (InputMismatchException e)
        {
            stdin.next();
            System.out.println("only use integers, please select again");
        }
        return chosenMenu;
    }

    /**
     * User enters amount of numbers for set A
     * Amount A the quantity of amount A for set
     * returns the number entered by user for set A
     */
    static int EnterAAmount(int amountA)
    {
        while(true)
        {
            try
            {
                System.out.printf("Please enter amount of numbers for set A: ");
                amountA = stdin.nextInt();
                return amountA;
            }
            catch (InputMismatchException e)
            {
                stdin.next();
                System.out.println();
                System.out.println("Invalid character, please use numbers");
            }
        }
    }

    /**
     * User enters amount of numbers for set B
     * @param amountB the quantity of amount B for set
     * @return returns the number entered by user for set B
     */
    static int EnterBAmount(int amountB)
    {
        while(true)
        {
            try
            {

```

```

        System.out.printf("Please enter amount of numbers for set B: ");
        amountB = stdin.nextInt();
        return amountB;
    }
    catch (InputMismatchException e)
    {
        stdin.next();
        System.out.println();
        System.out.println("Invalid Character, please use integer");
    }
}

/**
 * Functon to Enter set A values
 * Amount and chosen numbers
 * Returns SetA Values
 */
static int[] SetACount(int count, int setAmount, int[] setA)
{
    System.out.println();
    System.out.println("Set A: ");
    setA = new int [setAmount];
    for (count=0; count < setAmount; count++)
    {
        try
        {
            System.out.printf("Select Numbers: ");
            setA[count] = stdin.nextInt();
        }
        catch (InputMismatchException e)
        {
            stdin.next();
            count--;
            System.out.println("Invalid character please use numbers");
        }
    }
}
return setA;
}

/**
 * Functon to Enter set B values
 * Amount and chosen numbers
 * Returns Set B Values
 */
static int[] SetBCount(int count, int setBAmount, int[] setB)
{
    System.out.println();
    System.out.println("Set B: ");
    setB = new int [setBAmount];
    for (count=0; count < setBAmount; count++)
    {
        try
        {
            System.out.printf("Select Numbers: ");
            setB[count] = stdin.nextInt();
        }
        catch (InputMismatchException e)
        {
            stdin.next();
            count--;
            System.out.println("Invalid character please use numbers");
        }
    }
}
return setB;
}

/**
 * Function to perform chosen menu option
 */
static void PerformFuntion(int chosenMenu,int count,int countA, int countB, int setAmount,int setBAmount, int[] setA,
int[] setB,boolean check)
{
    int choice;
    boolean end = false;

    while(end == false)
    {
        chosenMenu = ChooseFunction();

        switch (chosenMenu)
        {
            case 1:
                //System.out.printf("%n");
                DisplaySets(count, setAmount, setBAmount, setA, setB);
                break;

            case 2:
                //System.out.printf("%2n");
                MaxNumberA(count, setAmount, setBAmount, setA);
                MaxNumberB(count, setAmount, setBAmount, setB);
                MinNumberA(count, setAmount, setBAmount, setA);
                MinNumberA(count, setAmount, setBAmount, setB);
                break;

            case 3:

```

```

        System.out.printf("Calculated the averages\n\r");
        Averages(count, setAAmount, setBAmount, setA, setB);
        break;

    case 4:
        //System.out.printf("%d");
        Overlap(count, setAAmount, setBAmount, setA, setB);
        break;

    case 5:
        //System.out.printf("%d");

        Intersection(countA, countB, setAAmount, setBAmount, setA, setB);
        System.out.println();
        break;

    case 6:
        //System.out.printf("%d");
        SetDifference(countA, countB, setAAmount, setBAmount, setA, setB);
        break;

    case 7:

        end = CheckEnd(end, check = false);
        break;

    default:
        System.out.printf("No Value");
        break;
    }
}

/**
 * Display Numbers in each Set Function
 */
static void DisplaySets(int count, int setAAmount, int setBAmount, int[] setA, int[] setB)
{
    System.out.println("Set A number are: ");
    for (count = 0; count < setAAmount; count++)
    {
        System.out.printf("%d ", setA[count]);
    }
    System.out.println();
    System.out.println("Set B number are: ");
    for (count = 0; count < setBAmount; count++)
    {
        System.out.printf("%d ", setB[count]);
    }
    System.out.println();
}

/**
 * Function to work out averages
 */
static void Averages(int count, int setAAmount, int setBAmount, int[] setA, int[] setB)
{
    int totalA=0, totalB=0;

    System.out.println("Set A Average of numbers is:");
    for( count=0; count < setAAmount; count++)
    {
        totalA = totalA + setA[count];
    }
    System.out.printf("%d ", totalA/setAAmount);
    System.out.println();
    System.out.println("Set B Average of numbers is:");
    for( count=0; count < setBAmount; count++)
    {
        totalB = totalB + setB[count];
    }
    System.out.printf("%d ", totalB/setBAmount);
}

/**
 * enters function to quit program and end system
 */
static boolean CheckEnd(boolean end, boolean check)
{
    int endAnswer = 0;

    while (check == false)
    {
        try
        {
            System.out.println("Are you sure you want to End");
            System.out.println("1 for yes");
            System.out.println("2 for No");
            endAnswer = stdin.nextInt();

            if (endAnswer == 1) {
                check = true;
                end = true;
            } else if (endAnswer == 2) {
                check = true;
                end = false;
            } else {
                System.out.println("Please try again");
                check = false;
            }
        }
    }
}

```

```

        } catch (InputMismatchException e)
        {
            stdin.next();
            System.out.println("Invalid Choice please use numbers");
        }
    }
    return end;
}

/**
 * Function to find Maximum number with Set A numbers
 */
static int MaxNumberA(int count, int setAAmount, int setBAmount, int[] setA)
{
    int maxA = setA[0];

    for (count = 0; count < setAAmount; count++) {
        if (setA[count] > maxA) {
            maxA = setA[count];
        }
    }
    System.out.println("The maximum number of set A is: " + maxA);

    return maxA;
}

/**
 * Function to find Maximum number with Set B numbers
 */
static int MaxNumberB(int count, int setAAmount, int setBAmount, int[] setB)
{
    int maxB = setB[0];

    for (count=0; count < setBAmount; count++)
    {
        if (setB[count] > maxB)
        {
            maxB = setB[count];
        }
    }
    System.out.println("The maximum number of set B is: " + maxB);

    return maxB;
}

/**
 * Function to find Minimum number with Set A numbers
 */
static int MinNumberA(int count, int setAAmount, int setBAmount, int[] setA)
{
    int minA = setA[0];

    for (count = 0; count < setAAmount; count++) {
        if (setA[count] < minA) {
            minA = setA[count];
        }
    }
    System.out.println("The minimum number of set A is: " + minA);

    return minA;
}

/**
 * Function to find Minimum number with Set B numbers
 */
static int MinNumberB(int count, int setAAmount, int setBAmount, int[] setB)
{
    int minB = setB[0];

    for (count=0; count < setBAmount; count++)
    {
        if (setB[count] < minB)
        {
            minB = setB[count];
        }
    }
    System.out.println("The minimum number of set B is: " + minB);

    return minB;
}

/**
 * Function to find if there is an overlap
 */
static void Overlap(int count, int setAAmount, int setBAmount, int[] setA, int[] setB)
{
    int minA = setA[count];
    int maxA = setA[count];
    int minB = setB[count];
    int maxB = setB[count];

    minA = MinNumberA(count, setAAmount, setBAmount, setA);
    maxA = MaxNumberA(count, setAAmount, setBAmount, setA);
    minB = MinNumberB(count, setAAmount, setBAmount, setB);
    maxB = MaxNumberB(count, setAAmount, setBAmount, setB);

    if ((minA < maxB) && (minB < maxA))
    {
        System.out.println("Overlap Found!");
    }
    else

```

```

        {
            System.out.println("No Overlap Found!");
        }
    }

    /**
     * Function to find if there is an intersection numbers
     */
    static void Intersection(int countA, int countB, int setAAmount, int setBAmount, int[] setA, int[] setB)
    {
        System.out.println();
        System.out.printf("The intersection is: ");
        for (countA = 0; countA < setAAmount; countA++)
        {
            for (countB = 0; countB < setBAmount; countB++)
            {
                if (setA[countA] == setB[countB])
                {
                    System.out.printf(" %d ", setA[countA]);
                }
            }
        }
    }

    /**
     * Function to find if there is a set difference numbers
     */
    static void SetDifference(int countA, int countB, int setAAmount, int setBAmount, int[] setA, int[] setB)
    {
        for (countA = 0; countA < setAAmount; countA++)
        {
            for (countB = 0; countB < setBAmount; countB++)
            {
                if (setA[countA] != setB[countB])
                {
                    System.out.printf(" %d ", setA[countA]);
                }
            }
        }
    }
}

```

4.2 Appendix 2 Part 2 coding

```

package MScSoftwareDevelopment;
import java.util.*;

public class Connect4
{
    public static String playerOneName;
    public static String playerTwoName;

    private static Scanner stdin = new Scanner(System.in);

    public static void main(String[] args)
    {
        PlayerNames();
        String[][] count = CreateDisplay();
        PrintDisplay(count);
        GameInit(count);
    }

    /**
     * Function to initialise connect 4 game
     * Checks which players turn to go.
     * prints Display
     * and enters win checker function
     */
    static void GameInit(String[][] count)
    {
        boolean loop = true;
        int total = 0;

        while(loop)
        {
            if (total % 2 == 0)
            {
                DropRed(count, playerOneName);
            }
            else
            {
                DropYellow(count, playerTwoName);
            }
            total++;

            PrintDisplay(count);

            if (CheckWin(count) != null)
            {
                if (CheckWin(count) == "R")
                {
                    System.out.println(playerOneName + " Wins");
                    loop = false;
                }
            }
        }
    }
}

```

```

        else if (CheckWin(count) == "Y")
        {
            System.out.println(playerTwoName + " Wins");
            loop = false;
        }
    }
}

/**
 * Function to select players Names
 */
public static void PlayerNames()
{
    System.out.println("Player 1: Please enter Name: ");
    playerOneName = stdin.next();
    //System.out.println(playerOneName); //Test

    System.out.println("Player 2: Please enter name: ");
    playerTwoName = stdin.next();
    //System.out.println(playerTwoName); //Test
}

/**
 * Function to Create display
 * 2D Array creation.
 */
private static String[][] CreateDisplay()
{
    int i,j;
    String[][] count = new String[7][15];

    for (i=0; i < count.length; i++)
    {
        for (j=0; j < count[i].length; j++)
        {
            if (j%2 == 0)
            {
                count[i][j] = "|";
            }
            else
            {
                count[i][j] = " ";
            }
            if (i == 6)
            {
                count[i][1] = "0";
                count[i][3] = "1";
                count[i][5] = "2";
                count[i][7] = "3";
                count[i][9] = "4";
                count[i][11] = "5";
                count[i][13] = "6";
            }
        }
    }
    return count;
}

/**
 * Function to Print and display the game grid
 */
static void PrintDisplay(String[][] count)
{
    int i,j;

    for (i =0; i<count.length; i++)
    {
        for (j=0; j<count[i].length; j++)
        {
            System.out.print(count[i][j]);
        }
        System.out.println();
    }
}

/**
 * Function for Player one to drop Red in column between 0-6
 * Player name string passed through
 */
static void DropRed(String[][] count, String PlayerOneName)
{
    int convert = 0;
    int i;
    boolean correctInput = false;

    while (!correctInput)
    {
        try
        {
            System.out.println(PlayerOneName + " Please Drop Red in column (0-6)");
            convert = 2 * stdin.nextInt() + 1;

            if (convert < 14)
            {
                correctInput = true;
            }
            else
            {
                System.out.println("Invalid column");
            }
        }
        catch (Exception e)
        {
            //
        }
    }
}

```

```

        catch (InputMismatchException e)
        {
            stdin.next();
            System.out.println("Invalid column");
        }
    }

    for (i=5; i>=0; i--)
    {
        if (count[i][convert] == " ")
        {
            count[i][convert] = "R";
            break;
        }
    }
}

/**
 * Function for Player one to drop Yellow in column between 0-6
 * Player name string passed through
 */
static void DropYellow (String[][] count, String PlayerTwoName)
{
    int convert = 0;
    int i;
    boolean correctInput = false;

    while (!correctInput)
    {
        try
        {
            System.out.println(PlayerTwoName + " Please Drop Yellow in column (0-6)");
            convert = 2 * stdin.nextInt() + 1;

            if (convert < 14)
            {
                correctInput = true;
            }
            else
            {
                System.out.println("Invalid column");
            }
        }
        catch (InputMismatchException e)
        {
            stdin.next();
            System.out.println("Invalid column");
        }
    }

    for (i=5; i>=0; i--)
    {
        if (count[i][convert] == " ")
        {
            count[i][convert] = "Y";
            break;
        }
    }
}

/**
 * Function to check win in all directions
 */
static String CheckWin(String[][] count)
{
    int i;
    int j;

    for (i = 0; i < 6; i++)
    {
        for (j = 0; j < 7; j += 2)
        {
            if ((count[i][j + 1] != " ") && (count[i][j + 3] != " ") && (count[i][j + 5] != " ")
                && (count[i][j + 7] != " ") && ((count[i][j + 1] == count[i][j + 3])
                && (count[i][j + 3] == count[i][j + 5]) && (count[i][j + 5] == count[i][j + 7])))
            {
                return count[i][j + 1];
            }
        }
    }

    for (i = 1; i < 15; i += 2)
    {
        for (j = 0; j < 3; j++)
        {
            if ((count[j][i] != " ") && (count[j + 1][i] != " ") && (count[j + 2][i] != " ")
                && (count[j + 3][i] != " ") && ((count[j][i] == count[j + 1][i])
                && (count[j + 1][i] == count[j + 2][i]) && (count[j + 2][i] == count[j + 3][i])))
            {
                return count[j][i];
            }
        }
    }

    for (i = 0; i < 3; i++)
    {
        for (j = 1; j < 9; j += 2)
        {
            if ((count[i][j] != " ") && (count[i + 1][j + 2] != " ") && (count[i + 2][j + 4] != " ")
                && (count[i + 3][j + 6] != " ") && ((count[i][j] == count[i + 1][j + 2])
                && (count[i + 1][j + 2] == count[i + 2][j + 4]) && (count[i + 2][j + 4] == count[i + 3][j + 6])))
            {

```

```

        {
            return count[i][j];
        }
    }
}

for (i = 0; i < 3; i++)
{
    for (j = 7; j < 15; j += 2)
    {
        if ((count[i][j] != " ") && (count[i + 1][j - 2] != " ") && (count[i + 2][j - 4] != " ")
            && (count[i + 3][j - 6] != " ") && ((count[i][j] == count[i + 1][j - 2])
            && (count[i + 1][j - 2] == count[i + 2][j - 4]) && (count[i + 2][j - 4] == count[i + 3][j - 5])))
        {
            return count[i][j];
        }
    }
}

return null;
}
}

```

James Jones Programming Assignment

GRADEMARK REPORT

FINAL GRADE

61 / 100

GENERAL COMMENTS

Instructor

PART ONE

=====

Engagement with Literature Skills (25) 19

Knowledge and Understanding Skills (10) 7

Cognitive and Intellectual Skills (10) 6

Transferable Skills (5) 4

Practical Application Skills (50) 25

Part 1 Software Testing

As a general overview of the testing process at the DVLA this report is fine, but it lacks detail. I would like to have seen more analysis of the tools used.

Code Part1 Number Sets

- a. Display numbers in sets COMPLETE
- b. Determine max and min FAILED
- c. Calculate average COMPLETE
- d. Determine overlap COMPLETE
- e. Intersection COMPLETE
- f. Set difference COMPLETE

Use of JUnit NO

Use of Git YES

Use of Java Doc LIMITED USAGE

The program crashed when displaying min/max, obviously needed some testing. Setting difference is incorrect.

PART TWO

=====

Engagement with Literature Skills (25) **13**

Knowledge and Understanding Skills (10) **6**

Cognitive and Intellectual Skills (10) **5**

Transferable Skills (5) **4**

Practical Application Skills (50) **25**

Part 2 Design Principles

Lacking detail in key areas. Noticeably, you give a lot of brief descriptions of the different kinds of cohesion, but at a resultant loss to a deeper description of what cohesion is.

Far too great a reliance on non-credible web sources.

Code Part 2 Connect 4

Display board COMPLETE

Enter column COMPLETE

Check column position is empty COMPLETE

Drop counter to position COMPLETE

Check win/lose COMPLETE

Blitz feature NOT IMPLEMENTED

Time bomb feature NOT IMPLEMENTED

Use of JUnit NO

Use of Git YES

Use of Java Doc LIMITED USAGE

Code doesn't detect when the column is full. Be careful when comparing strings, the "==" operator doesn't work the way you expect use `string1.equals(string2)` instead.

Overall Comments

Used the same repository for both code parts and even used the same branch. Should at least used different branches for both parts. No unit testing. A number of errors in the code.

Marks Part One: **61**

Marks Part Two: **53**

PAGE 1

PAGE 2

PAGE 3

PAGE 4



Comment 1

Define acronym before using it.

System Unit Test?



Comment 2

Application Unit Test?



Comment 3

What do these do?

