

SOFTWARE ENGINEERING PROJECT

Group 1:

Mobina Tooranisama

Abdul Hafeez Mohammad

Simrah Azfar

Lubna Al Rifaie

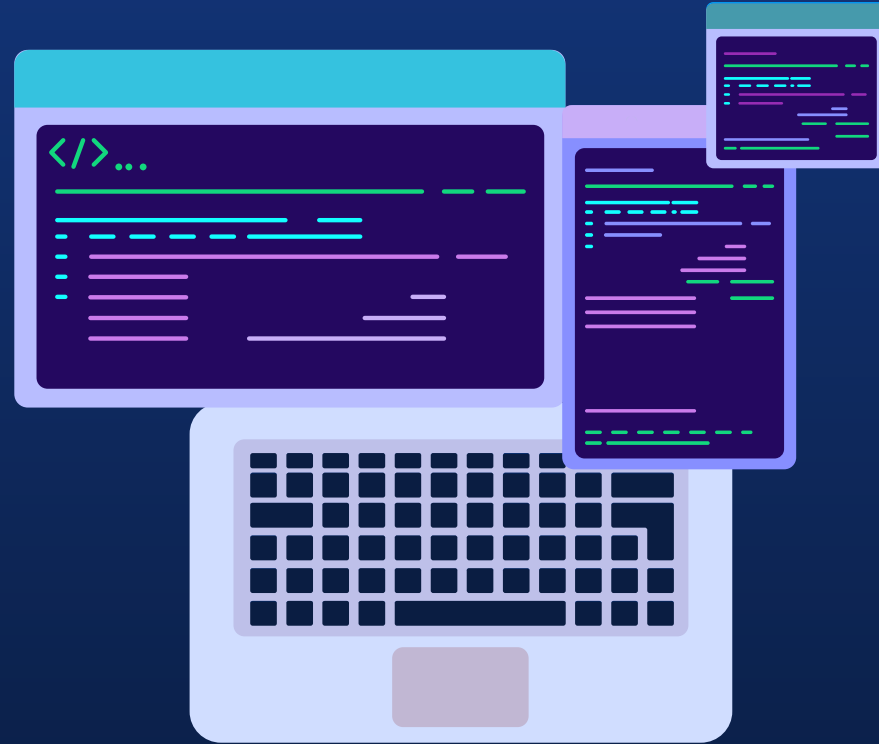


TABLE OF CONTENTS

01

Project Overview

02

Requirements Analysis

03

Software Design

04

Implementation

05

Error Testing & Analysis

06

Deployment/Demonstration





01

PROJECT OVERVIEW





Project Overview

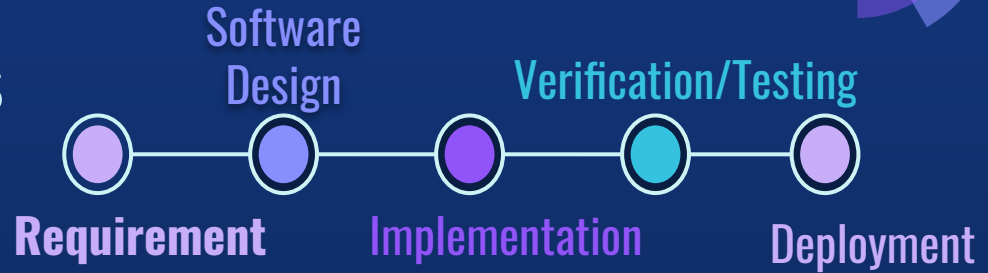
- Design and develop a software application to read two text files and format the data output to a new file
- **Note 1:** The student ID, student name, course code, and the grades will be separated with comma in the input files.
- **Note 2:** The student ID and students' name are unique. However, one student may take multiple courses.





Engineering design process


Waterfall Process Model



Why?

- The requirements are precisely known in advance
- The requirements include no unresolved high-risk items
- The requirements will not change much during development





02

REQUIREMENTS ANALYSIS



REQUIREMENTS

The two inputted files will adhere to the format below.

≡ CourseFile.txt

```
1 306851690, CP460, 74, 98, 76, 52
2 413787382, CP164, 66, 82, 81, 75
3 454730171, CP414, 69, 80, 72, 87
4 620731804, MA200, 63, 82, 58, 68
```

CourseFile.txt:

- Student ID
- Course Code
- Test 1
- Test 2
- Test 3
- Final Exam

≡ NameFile.txt

```
1 237711821, Saara Beattie
2 113820457, Emre Lowry
3 642176077, Teddy Hyde
4 413787382, Markus Cote
```

NameFile.txt:

- Student ID (max length 9 bytes)
- Student Name (max length 20 bytes)

REQUIREMENTS

Continued...

The output file will adhere to the format below.

```
≡ output.txt
1  STUDENT-ID  NAME                COURSE  GRADE
2  113820457,  Emre Lowry,        ST494,  68.6
3  113820457,  Emre Lowry,        ST259,  62.4
4  119616823,  Mehreen Franks,    ST494,  72.2
5  119616823,  Mehreen Franks,    CP460,  65.6
6  150295440,  Abdurrahman Person, ST259,  84.2
7  150295440,  Abdurrahman Person, BI111,  77.4
8  172348496,  Maggie Barnard,    EC120,  75.0
9  172348496,  Maggie Barnard,    MA200,  64.2
10 185598260,  Reilly Woodward,   BU111,  78.0
11 185598260,  Reilly Woodward,   BU352,  86.6
12 229922406,  Zaynah Alfaro,     EC120,  79.0
13 229922406,  Zaynah Alfaro,     CP317,  65.4
```

Output.txt:


- Student ID (max length 9 bytes)
- Student Name (max length 20 bytes)
- Course Name (5 bytes)
- Overall Grade (max 2 bytes)



FURPS!

Categorizing the system requirements

Functionality	Usability	Reliability	Performance	Supportability
<ul style="list-style-type: none">- Read student information files- Calculate Final Grades- Output formatted version of data	<ul style="list-style-type: none">- GUI not needed- Well documented code- Easy to read output data	<ul style="list-style-type: none">- Advanced error detection and notification- Zero tolerance for user errors ensures no crashing	<ul style="list-style-type: none">- Efficient code- Minimum resource consumption- Quick response time	<ul style="list-style-type: none">- Maintainable- Multi-Platform Compatible- Adaptable





Prioritizing Requirements

MOSCOW method

MUST HAVE: Functional software which uses object oriented programming to read and calculate two input files with the names nameFile.txt and courseFile.txt and create a formatted output file

SHOULD HAVE: Error handling to assure the validity of the inputted files and the students, courses and grades requirements. Flexibility to change the formatting of the output file.

COULD HAVE: A database to store the student's data. A graphical user interface

WON'T HAVE: Spaghetti Code(disorganized, unstructured and hard to maintain code)

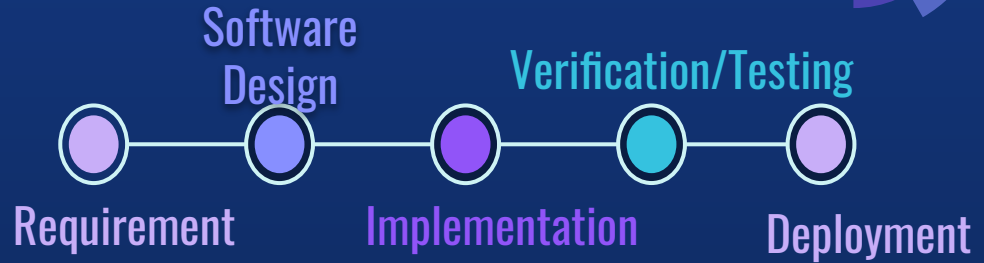


03

Software Design



High Level Software Design



Component Based Architectural Design

A software object, intended to interact with other components, encapsulating certain functionality or a set of functionalities

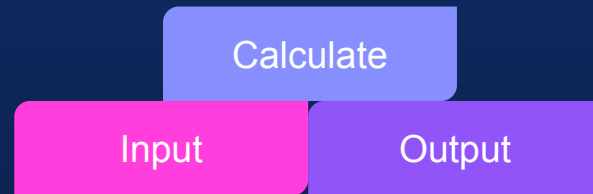
Focuses on the decomposition of the design into individual functional or logical components that represent well-defined communication interfaces containing methods, events, and properties.



Software Architectural Design

Three Major Components:

- Input Data Manipulation
- Grade Calculation
- Data Output

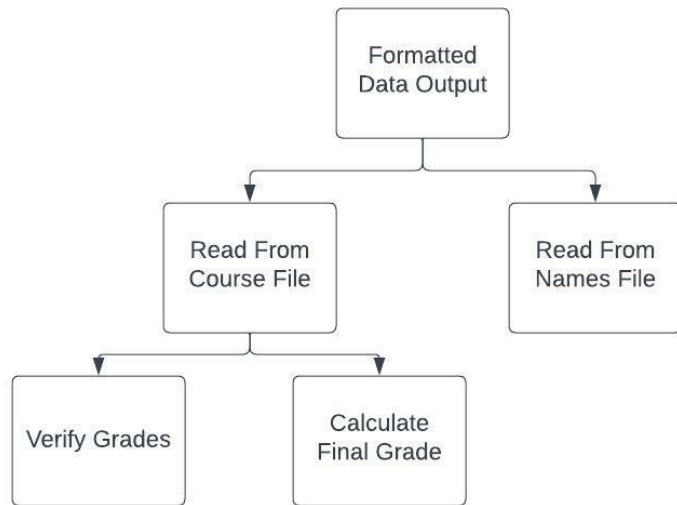




High Level Software Design

Decomposition Design Principle

Breaking a complex problem or system into parts that are easier to conceive, understand, program, build, and maintain





Software Low-Level (Detailed) Design

Interface Design



GUI design



Internal Component
Design



Data Design

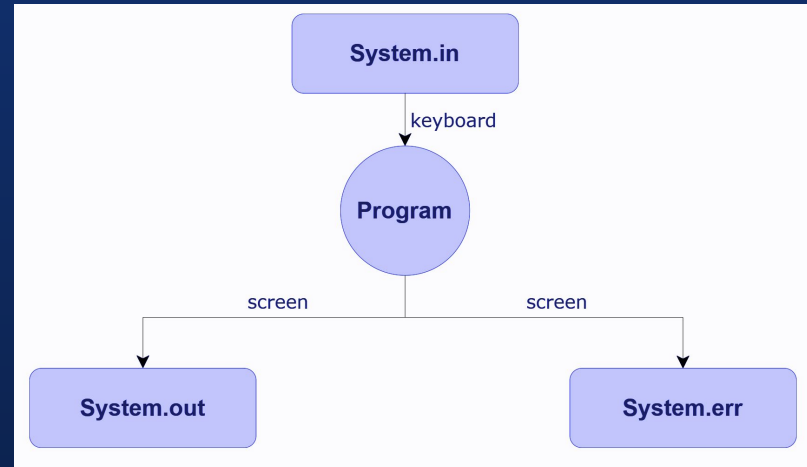




Low-Level (Detailed) Software Design

Interface (Internal/External) Design

A visual part of computer application or operating system through which a user interacts with a computer or a software system

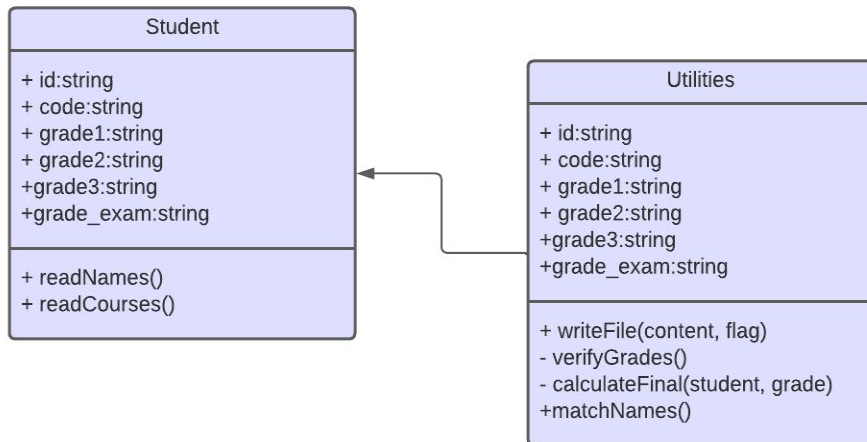


The role of the user interface is to transmit data back and forth between the user and the program.



Internal Component Design

UML structure diagram - Class Diagram





04

Implementation





Implementation

Types of Object Oriented Programming(OOP) Features used

- Inheritance
- Encapsulation
- Polymorphism





Implementation

Inheritance

Utilities class is an extension of the Student class and will inherit methods and attributes from the student parent class,

```
public class utilities extends student{  
  
    private static String outputFile = "output.txt";  
  
    utilities(String id, String code, String grade1, String grade2, String grade3, String grade_exam) {  
        super(id, code, grade1, grade2, grade3, grade_exam);  
    }  
}
```





Implementation

Encapsulation

Student class will hide student information through the use of private variables and Getters and Setters. There is no direct references to data without calling a Getter method.

```
public String getName(){  
    return NAME;  
}
```

```
public String getCode(){  
    return CODE;  
}
```

```
public String getID(){  
    return ID;  
}
```

```
private static File coursesFile = new File(pathname: "CourseFile.txt");  
private static File namesFile = new File(pathname: "NameFile.txt");
```





Implementation

Polymorphism

The student class uses two constructors known as constructor overloading which is an example of compile time polymorphism

```
student(String id, String code, String grade1, String grade2, String grade3, String grade_exam){  
    this.ID = id;  
    this.CODE = code;  
    this.GRADE1 = grade1;  
    this.GRADE2 = grade2;  
    this.GRADE3 = grade3;  
    this.GRADE_EXAM = grade_exam;  
}  
  
student(String id, String name){  
    this.ID = id;  
    this.NAME = name;  
}
```





Grade Calculation

First verify the grades of the students and ensure there are no errors

For example:

- no grades outside of 0-100 range
- Grades are all numerical, do not include letters

```
private static void calculateFinal(student student, double[] grade){
    double finalGrade = (grade[0]*0.2) + //grade 1 times 20% weightage
                        (grade[1]*0.2) + //grade 2 times 20% weightage
                        (grade[2]*0.2) + //grade 3 times 20% weightage
                        (grade[3]*0.4); //exam grade times 40% weightage summed with test grades
    int scale = (int) Math.pow(a: 10, b: 1); //scale is needed to accurately round to 1 decimal place

    finalGrade = (double) Math.round(finalGrade * scale)/scale; //return a rounded value

    student.setFinal(Double.toString(finalGrade));
}
```





Grade Calculation Verification

STUDENT-ID	NAME	COURSE	GRADE
113820457	Emre Lowry	ST494	68.6
113820457	Emre Lowry	ST259	62.4

Grades From Data Output

113820457, ST494, 70, 95, 54, 62

113820457, ST259, 93, 54, 51, 57

Grades From Data Input

ST494 Final Grade Calculation:

$$\begin{array}{ccccccc} (70 \times 0.20) & + & (95 \times 0.20) & + & (54 \times 0.20) & + & (62 \times 0.4) = 68.6 \\ 14 & & 19 & & 10.8 & & 24.8 = 68.6 \end{array}$$



ST259 Final Grade Calculation:

$$\begin{array}{ccccccc} (93 \times 0.20) & + & (54 \times 0.20) & + & (51 \times 0.20) & + & (57 \times 0.4) = 62.4 \\ 18.6 & & 10.8 & & 10.2 & & 22.8 = 62.4 \end{array}$$



**Final grades are
correctly calculated**



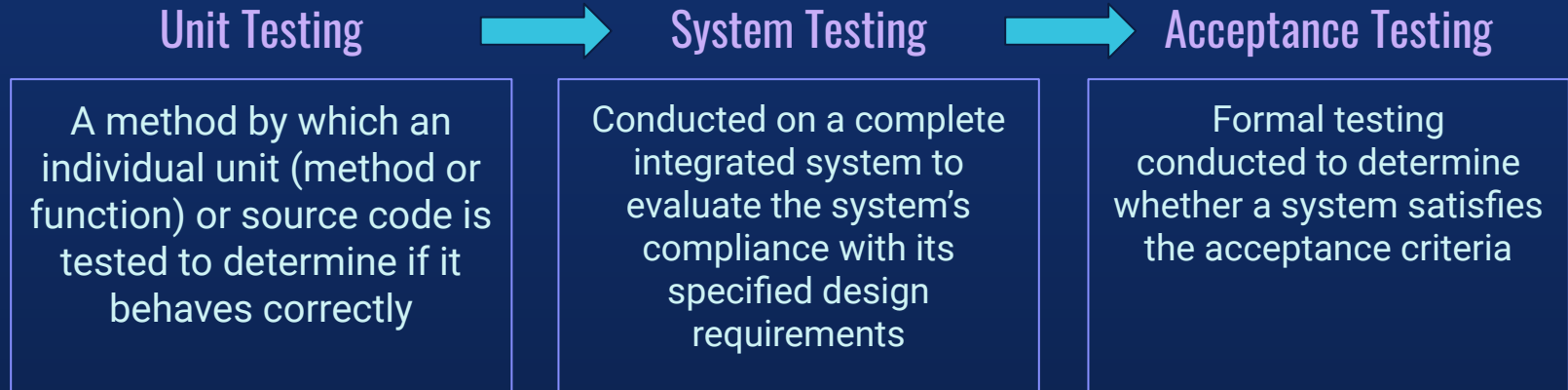


05

Testing & Error Analysis



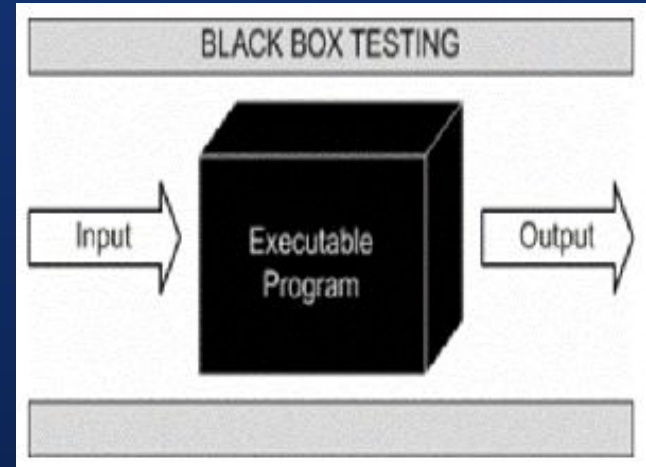
Software Testing Plan



Software Testing Technique

Black Box Testing

- A method of software testing that examines the functionality of the software without peering into its internal structures or workings
- Can apply to all test levels from unit testing to system testing



Software Testing Error Handling

Offensive Programming Strategy


Upon error detection, the software displays an error message and stops running. There is zero tolerance for user error

```
String msg = "ERROR: on line " + counter + " of NameFile.txt"; //generate an error message with line number
utilities.writeFile(msg, flag: true); //submit error message with True flag
System.exit(status: 1); //stop running code
```

Error Analysis

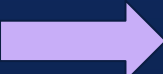
Example of error where
studentID are mismatched
between files

NameFile.txt



9	440254806,	Bethanie Oliver
10	422078053,	Deon Haynes
11	510661066,	Ebrahim Lloyd
12	620731864,	Rudy Begum
13	500461394,	Emily Hale
14	580228531,	Lincoln Bowler
15	185598260,	Reilly Woodward
16	241905753,	Emme Blackmore

CourseFile.txt



1	306851690,	CP460,	74,	98,	76,	52
2	413787382,	CP164,	66,	82,	81,	75
3	454730171,	CP414,	69,	80,	72,	87
4	620731804,	MA200,	63,	82,	58,	68
5	185598260,	BU111,	58,	98,	56,	89
6	150295440,	ST259,	66,	84,	95,	88
7	241905753,	ST260,	59,	69,	56,	96

≡ output.txt

```
1 ERROR: Student ID 620731804 in CourseFile.txt does not match with any ID's in NameFile.txt
```



Demonstration!



Que

Any Questions?