



Student Name	OSAMA MOHAMMED ZIAD HASAN
Student ID	23110709
HTU Course Number and Title	40201100 Programming
BTEC Unit Number and Title	H/618/7388 Programming
Academic Year	2023-2024 Fall
Assignment Author	Sultan Mahmoud Ahmed Al-Rashdan
Assignment Title	Programming using Java
Submission Date	09/6/2024

Table of Contents

Part I:	<i>Error! Bookmark not defined.</i>
process of building an application.	<i>Error! Bookmark not defined.</i>
The concept of algorithm	<i>Error! Bookmark not defined.</i>
Ensuring that the algorithm is efficient and effective	<i>Error! Bookmark not defined.</i>
Steps for writing code	<i>Error! Bookmark not defined.</i>
Process of compiling code into machine-readable	<i>Error! Bookmark not defined.</i>
Part II:	<i>Error! Bookmark not defined.</i>
Defining the procedural programming paradigm	<i>Error! Bookmark not defined.</i>
The concept of object-oriented programming (OOP)	<i>Error! Bookmark not defined.</i>
Describing the event-driven programming paradigm	<i>Error! Bookmark not defined.</i>
Examples of code demonstrates the three paradigms	<i>Error! Bookmark not defined.</i>
Compare between paradigms	<i>Error! Bookmark not defined.</i>
The scenario code	9
To ensure good code quality	<i>Error! Bookmark not defined.</i>
Importance of testing code before execution	<i>Error! Bookmark not defined.</i>
Part III:	<i>Error! Bookmark not defined.</i>
Steps to create a new project in IDE	<i>Error! Bookmark not defined.</i>
Features and tools in the IDE	<i>Error! Bookmark not defined.</i>
IDE debugging features	<i>Error! Bookmark not defined.</i>
Comparing between using an IDE and not using one	<i>Error! Bookmark not defined.</i>
Part IV:	<i>Error! Bookmark not defined.</i>
Discuss error types	<i>Error! Bookmark not defined.</i>
Steps in the debugging process	<i>Error! Bookmark not defined.</i>
The role of breakpoints	<i>Error! Bookmark not defined.</i>
Security vulnerabilities	<i>Error! Bookmark not defined.</i>
My experiences with the debugging	<i>Error! Bookmark not defined.</i>
Part V:	<i>Error! Bookmark not defined.</i>
Coding standards	<i>Error! Bookmark not defined.</i>
Naming conventions	<i>Error! Bookmark not defined.</i>
Importance of having a coding standard for a development team	<i>Error! Bookmark not defined.</i>

References: *Error! Bookmark not defined.*

Part 1:

1. To build an application, there are some basic steps that we must follow to get the best result in the end, which are:

Step 1: Define and understand the goal or the problem.

Step 2: Find a way to solve the problem or to achieve the goal and choose the best way.

Step 3: Preset the chosen method we used in simple steps known as “algorithm”, which represents the processes and outputs that we have achieved.

Step 4: Writing a code that implements the steps of the algorithm.

Step 5: Test the code.

Step 6: Document the code.

2. Algorithms, in absolute, are the ordered steps that are used to reach a specific goal. They are used in many fields, most famously mathematics, and this is exactly how they are applied in programming. We arrange the steps that include inputs and processes that explain our approach to solving a problem or reaching a specific goal.

3. The ideal algorithm must fulfill these six conditions:

1. Clear and Unambiguous, so that every line in the algorithm is easy to understand and turn into a line of code.

2. Well-defined Inputs, so that the reader does not get confused while entering them.

3. Well-defined Outputs, so that the result matches the desired result of the algorithm.

4. Finiteness, so that the result is clear and does not have anything complicated in terms of the beginning and end of the code.

5. Feasible, the algorithm must be clear, and its application is realistically possible.

6. Language Independent. The algorithm must not be limited in its implementation in a particular programming language, so that it can be converted to any programming language the implementer wants.

To ensure these matters, they must be tested and given to more than one programmer, and to ensure that everyone applies them in the same way and with the same outputs and inputs to

ensure that they are easy and uncomplicated. The algorithm must be developable in the future and not limited or closed so that it can accommodate larger inputs when developing it, in addition to paying attention to its quality and ease of applying it quickly. If it includes all these conditions, the algorithm will have proven its impact and efficiency.

(Alfie, 2024)

4. To write a code using any programming language, there are steps that must be applied, which are:

Step 1: Choose a programming language to write the code using.

Step 2: Writing the code using text editor.

Step 3: Compile the code to convert it from a programming code into a into machine-readable instructions.

Step 4: Run the code to be interpreted into a desired output.

5. To compile a code after you finished writing it, when we click run, this file is converted into a binary file consisting of the numbers 0 and 1 only, and this is how the compiler can read the code, and this is the same file that is formed when we write `javac classname.java` in the command prompt if we are using the JDK, but in the IDE this is done automatically. In this way, the compiler can now read the code that we wrote by translating the binary file into output.

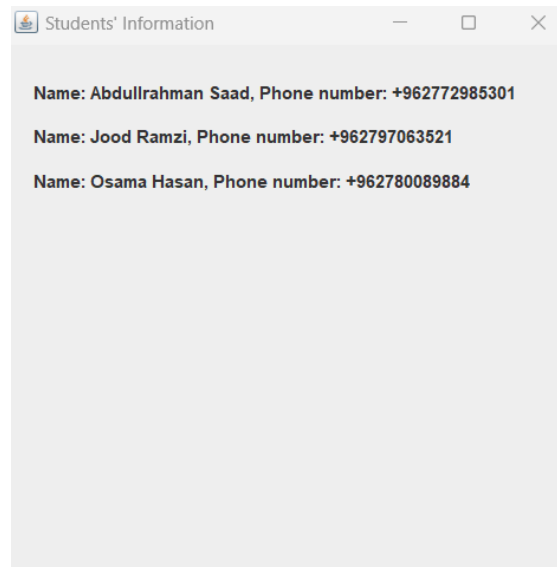
(Essence, 2017)

6.

```
//Arrange the students' names in the alphabetical order
private static void bubbleSort(ArrayList<Student> students) {
    int n = students.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - 1 - i; j++) {
            if (students.get(j).compareTo(students.get(j + 1)) > 0) {
                // Swap students[j] and students[j+1]
                Student temp = students.get(j);
                students.set(j, students.get(j + 1));
                students.set(j + 1, temp);
            }
        }
    }
}
```

- Here, I used bubble sort algorithm to order the names in the ArrayList alphabetically

The output:



Part 2:

1. The procedural programming paradigm is considered the cornerstone of the OOP system, and it works to make the machine read the code proceed in order in the sequence of commands, starting from the first line from top to bottom.

Here are some key characteristics of the procedural programming paradigm:

- Ease of navigating and changing some inputs for each stage of the code, as sometimes you need to change something after a certain stage of the code, so this paradigm facilitates this process.

- It makes it easy for anyone to read the code and make any modification or maintenance to it without complexity.

2. Object-oriented programming (OOP) is an object-based paradigm, where more than one class can be created in a single code, and each class refers to a specific object. One class has specific attributes and methods that distinguish it, and it can be the same in some different classes. Some fundamental principles and features of OOP:

- Classes
- Objects
- Encapsulation
- Inheritance
- Polymorphism

3. In event-driven programming, an graphical user interface (GUI) is used for obtaining user input instead of using the console interface, which is considered more effective than the console interface, and is what distinguishes it from Procedural and OOP paradigms.

4.

```
public class Person {
    private String name;
    private String phoneNumber;

    //Constructor
    public Person(String name, String phoneNumber) {
        this.name = name;
        this.phoneNumber = phoneNumber;
    }

    // Getter for name
    public String getName() {
        return name;
    }

    // Setter for name
    public void setName(String name) {
        this.name = name;
    }

    // Getter for phoneNumber
    public String getPhoneNumber() {
        return phoneNumber;
    }

    // Setter for phoneNumber
    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
}

//student registering
registerButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String name = nameField.getText();
        String phoneText = phoneField.getText();
        if (!name.isEmpty() && !phoneText.isEmpty()) {
            students.add(new Student(name, phoneText));
            JOptionPane.showMessageDialog(frame, "Student added successfully", "Success", JOptionPane.INFORMATION_
            coursesButton.setVisible(true);
        } else {
            JOptionPane.showMessageDialog(frame, "Please fill in all fields correctly", "Error", JOptionPane.E
        }
    }
});
```

The code goes line by line in the order of commands, which facilitates the coding process, which is the Procedural paradigm, we used the OOP, where we put the Getters, Setters and Constructors

in (Person) class and main code on another class, and we used the Event-Driven (GUI) as it is clear like the JoptionPane and others.

5.

- Procedural Programming Paradigm:

Implementing commands in order, which helps in understanding the method of writing the code for any reader of the code after writing it, and helps in writing the code in a better and smoother way and gives the code more flexibility, which helps the success of the application

```
//Courses menu
private static void Courses(Scanner scanner) {
    System.out.println("\nChoose the course you want to purchase:");
    String[] courses = { "1. Mathematics - Mr. Mohammed Ziad", "2. Chemistry - Mr. Tareq Mohammed",
        "3. Physics - Mr. Ahmed Al-tamimi", "4. Biology - Mr. Mousa Al-Zamil",
        "5. Arabic - Mr. Hassan Salhab" };

    for (int i = 0; i < courses.length; i++) {
        System.out.println(courses[i]);
    }
    System.out.print("Choose a course number: ");
    int courseChoice = scanner.nextInt();
    scanner.nextLine();
    if (courseChoice > 0 && courseChoice <= courses.length) {
        Payment(scanner);
    } else {
        System.out.println("Invalid course choice.");
    }
}
```

If the arrangement of the numbers that are in red, which is the order of the commands, changed, the output will change

- Object-Oriented Programming Paradigm:

When some methods, objects, constructors and others are in a class that differs from the main programming class, this gives us easy access to them and makes the distribution of orders to the team easier.

```
//inheritance
public class Student extends Person {

//Polymorphism (toString method)
public String toString() {
    return super.toString();
}
```

```
//Admin view
private static void Admin(Scanner scanner, ArrayList<Student> students) {
    System.out.print("Enter your admin pin: ");
    int pin = scanner.nextInt();
    scanner.nextLine();
    if (pin == 8471 || pin == 2801 || pin == 9452) {
        System.out.println("\nStudents' information:");
        for (int i = 0; i < students.size(); i++) {
            System.out.println(students.get(i).toString());
        }
    } else {
        System.out.println("Error: Incorrect PIN");
    }
}
```

Here, the toString method was written in a class, and then in the main class it has been used, this is a good example of the oop.

- Event-Driven paradigm:

In this paradigm, frames are used instead of console, which is considered much more effective for the user. The frame contains swing components and listeners, which makes it easier for the coder to write and facilitates the use process for the user.

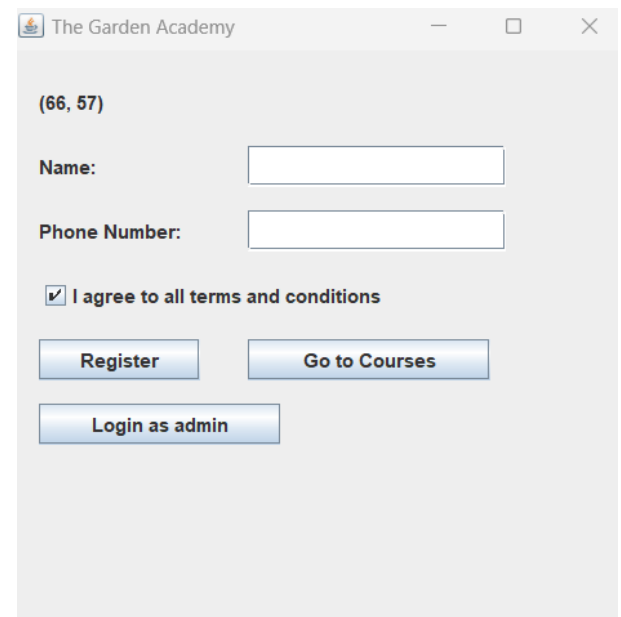
```
//menus (main menu) & inputing student's informations
JFrame frame = new JFrame("The Garden Academy");
frame.setSize(400, 400);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setLayout(null);
JLabel lbl = new JLabel("Welcome to the garden academy!");
lbl.setBounds(20, 20, 200, 25);
frame.add(lbl);
JLabel nameLabel = new JLabel("Name:");
nameLabel.setBounds(20, 60, 80, 25);
frame.add(nameLabel);

JTextField nameField = new JTextField();
nameField.setBounds(150, 60, 160, 25);
frame.add(nameField);

JLabel phoneLabel = new JLabel("Phone Number:");
phoneLabel.setBounds(20, 100, 120, 25);
frame.add(phoneLabel);

JTextField phoneField = new JTextField();
phoneField.setBounds(150, 100, 160, 25);
frame.add(phoneField);

JCheckBox chk = new JCheckBox("I agree to all terms and conditions");
chk.setBounds(20, 140, 250, 25);
frame.add(chk);
JButton registerButton = new JButton("Register");
registerButton.setBounds(20, 180, 100, 25);
frame.add(registerButton);
JButton coursesButton = new JButton("Go to Courses");
coursesButton.setBounds(150, 180, 150, 25);
coursesButton.setVisible(false);
frame.add(coursesButton);
JButton admin = new JButton("Login as admin");
admin.setBounds(20, 220, 150, 25);
frame.add(admin);
```



This interface is much more efficient than the console, so the user can navigate the application more smoothly and use the mouse

6. The garden academy is an educational academy composed of elite professors, which aims to raise the students of the Hashemite Kingdom of Jordan to a level that enables them to finish their studies with the best grades.

First, I made 3 different classes within the OOP, Person class, Student class and main class

Person class:

```
package myPackage;

public class Person {
    private String name;
    private String phoneNumber;

    //Constructor
    public Person(String name, String phoneNumber) {
        this.name = name;
        this.phoneNumber = phoneNumber;
    }

    // Getter for name
    public String getName() {
        return name;
    }

    // Setter for name
    public void setName(String name) {
        this.name = name;
    }

    // Getter for phone Number
    public String getPhoneNumber() {
        return phoneNumber;
    }

    // Setter for phoneNumber
    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    // toString method
    public String toString() {
        return "Name: " + name + ", Phone number: " + phoneNumber;
    }

    // CompareTo method to compare names (for the bubbleSort)
    public int compareTo(Person other) {
        return this.name.compareTo(other.name);
    }
}
```

- I putted the getters and setters for students' names and phone numbers
- I used toString method to use in the main class for presenting students' name and numbers

- I used compareTo method to compare students' names, to arrange them in alphabetical order in "main" Class

Student class:

```
//inheritance
public class Student extends Person {

    // Constructor
    public Student(String name, String phoneNumber) {
        super(name, phoneNumber);
    }

    // Polymorphism (Overload)
    public Student(String name) {
        super(name, "");
    }

    //Polymorphism toString method(Override)
    public String toString() {
        return super.toString();
    }
}
```

- I used inheritance to inherits methods from the class "Person"
- I made a constructor that takes 2 parameters (Student name, Phone number) and calls the constructor of the superclass (Person)
- I used overload
- I called the toString method from "Person" class, which is an override method

Main class:

```
public class Main {
    static ArrayList<Student> students = new ArrayList<>();

    public static void main(String[] args) {
        //menus (main menu) & inputting student's informations
        JFrame frame = new JFrame("The Garden Academy");
        frame.setSize(400, 400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);
        JLabel lbl = new JLabel("Welcome to the garden academy!");
        lbl.setBounds(20, 20, 200, 25);
        frame.add(lbl);
        JLabel nameLabel = new JLabel("Name:");
        nameLabel.setBounds(20, 60, 80, 25);
        frame.add(nameLabel);
        JTextField nameField = new JTextField();
        nameField.setBounds(150, 60, 160, 25);
        frame.add(nameField);
        JLabel phoneLabel = new JLabel("Phone Number:");
        phoneLabel.setBounds(20, 100, 120, 25);
        frame.add(phoneLabel);
        JTextField phoneField = new JTextField();
        phoneField.setBounds(150, 100, 160, 25);
        frame.add(phoneField);
        JCheckBox chk = new JCheckBox("I agree to all terms and conditions");
        chk.setBounds(20, 140, 250, 25);
        frame.add(chk);
        JButton registerButton = new JButton("Register");
        registerButton.setBounds(20, 180, 100, 25);
        frame.add(registerButton);
        JButton coursesButton = new JButton("Go to Courses");
        coursesButton.setBounds(150, 180, 150, 25);
        coursesButton.setVisible(false);
        frame.add(coursesButton);
        JButton admin = new JButton("Login as admin");
        admin.setBounds(20, 220, 150, 25);
        frame.add(admin);
    }
}
```

- The name and phone number must be inputted by the student
- The student cannot move to the courses page unless he has registered his name and phone number

```
//student registering
registerButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String name = nameField.getText();
        String phoneText = phoneField.getText();
        if (!name.isEmpty() && !phoneText.isEmpty()) {
            students.add(new Student(name, phoneText));
            JOptionPane.showMessageDialog(frame, "Student added successfully", "Success", JOptionPane.INFORMATION_MESSAGE);
            coursesButton.setVisible(true);
        } else {
            JOptionPane.showMessageDialog(frame, "Please fill in all fields correctly", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
});
```

- If at least one of the fields is empty, the user will be given an error message, but if they are full, a new "Student" object using the constructor will be created, and the data will be stored in the arraylist. After registering the student, the courses button will appear

```
nameField.addKeyListener(new KeyAdapter() {
    @Override
    public void keyTyped(KeyEvent e) {
        char c = e.getKeyChar();
        if (!Character.isLetter(c) && c != ' ') {
            e.consume();
        }
    }
});
```

- I used the keyListener to make the student cannot write a number or special character in the name field

```
frame.addMouseMotionListener(new MouseMotionAdapter() {
    @Override
    public void mouseMoved(MouseEvent e) {
        lbl.setText("(" + e.getX() + ", " + e.getY() + ")");
    }
});

frame.setVisible(true);
}
```

(130, 34)

- I used the keyListener to show the student where his mouse at in the frame, as an aesthetic addition

```

coursesButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Courses();
    }
});

//Courses menu
private static void Courses() {
    JFrame courseFrame = new JFrame("Courses");
    courseFrame.setSize(400, 400);
    courseFrame.setLayout(null);

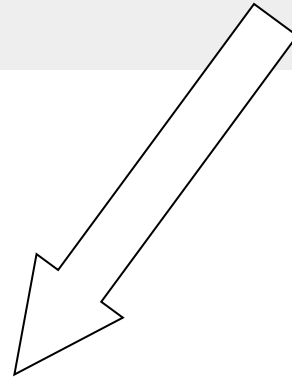
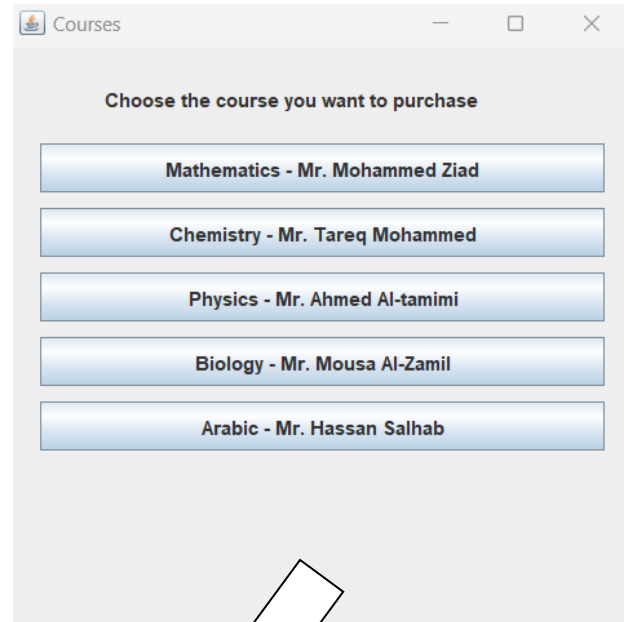
    JLabel courseLabel = new JLabel("Choose the course you want to purchase");
    courseLabel.setBounds(60, 20, 300, 25);
    courseFrame.add(courseLabel);

    //Courses
    String[] courses = {
        "Mathematics - Mr. Mohammed Ziad",
        "Chemistry - Mr. Tareq Mohammed",
        "Physics - Mr. Ahmed Al-tamimi",
        "Biology - Mr. Mousa Al-Zamil",
        "Arabic - Mr. Hassan Salhab"
    };

    int y = 60;
    for (int i = 0; i < courses.length; i++) {
        String course = courses[i];
        JButton courseButton = new JButton(course);
        courseButton.setBounds(20, y, 350, 30);
        courseButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Payment();
            }
        });
        courseFrame.add(courseButton);
        y += 40;
    }

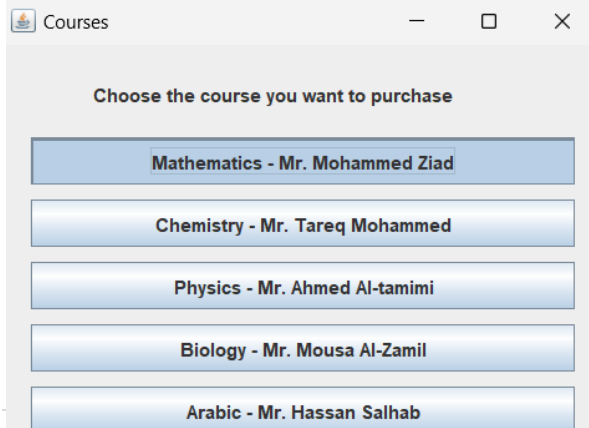
    courseFrame.setVisible(true);
}

```



- When pressing the courses button, you will move to a new page with all courses and the student chooses the course he wants

- The courses are all stored inside array, and each one is displayed in button by for loop, when you pressed on any course, it will move to the payment page.



```
//Payment page
private static void Payment() {
    JFrame paymentFrame = new JFrame("Payment");
    paymentFrame.setSize(400, 400);
    paymentFrame.setLayout(null);

    //Student enters his bank information
    JLabel paymentLabel = new JLabel("Please enter your bank information");
    paymentLabel.setBounds(20, 20, 200, 25);
    paymentFrame.add(paymentLabel);

    JLabel cardNumberLabel = new JLabel("Card number:");
    cardNumberLabel.setBounds(20, 60, 120, 25);
    paymentFrame.add(cardNumberLabel);
    JTextField cardNumberField = new JTextField();
    cardNumberField.setBounds(150, 60, 160, 25);
    paymentFrame.add(cardNumberField);

    JLabel cscLabel = new JLabel("CSC:");
    cscLabel.setBounds(20, 100, 100, 25);
    paymentFrame.add(cscLabel);
    JTextField cscField = new JTextField();
    cscField.setBounds(150, 100, 160, 25);
    paymentFrame.add(cscField);

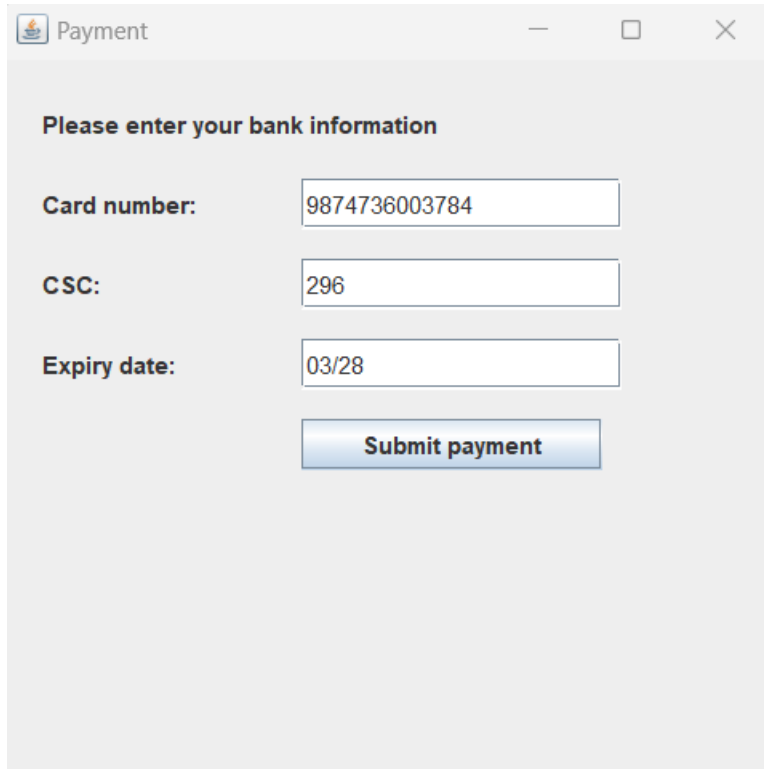
    JLabel expiryLabel = new JLabel("Expiry date:");
    expiryLabel.setBounds(20, 140, 120, 25);
    paymentFrame.add(expiryLabel);
    JTextField expiryField = new JTextField();
    expiryField.setBounds(150, 140, 160, 25);
    paymentFrame.add(expiryField);

    JButton submitPayment = new JButton("Submit payment");
    submitPayment.setBounds(150, 180, 150, 25);
    paymentFrame.add(submitPayment);

    JLabel paymentDone = new JLabel("");
    paymentDone.setBounds(20, 220, 300, 25);
    paymentFrame.add(paymentDone);

    //When finishing entering bank informations
    submitPayment.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            paymentDone.setText("Your payment was processed successfully.");
        }
    });

    paymentFrame.setVisible(true);
}
```



Payment

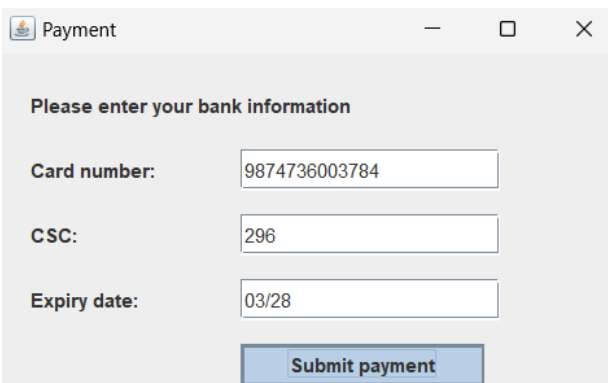
Please enter your bank information

Card number:

CSC:

Expiry date:

- When choosing the course that the student wants to buy, now it is time to pay, the student enters his banking information to buy the course



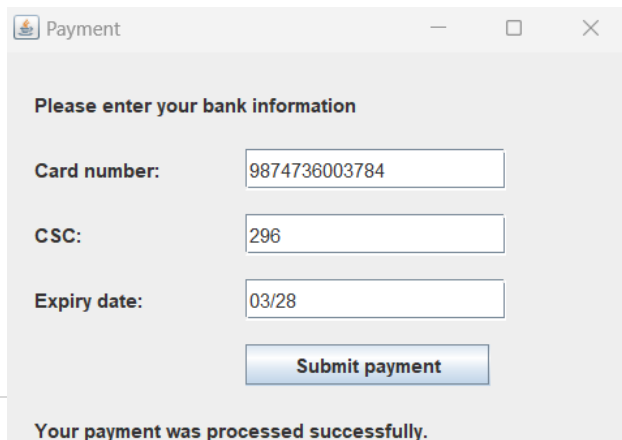
Payment

Please enter your bank information

Card number:

CSC:

Expiry date:



Payment

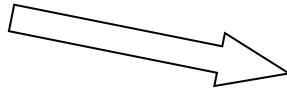
Please enter your bank information

Card number:

CSC:

Expiry date:

Your payment was processed successfully.



Well, what about employees in the academy (Admins)? There are 3 main admins who have powers in seeing student information (names and phone numbers)

The screenshot shows a Java Swing window titled "The Garden Academy". Inside, there's a registration form with the following elements: a label "(154, 247)" at the top; a "Name:" label followed by a text input field; a "Phone Number:" label followed by a text input field; a checked checkbox labeled "I agree to all terms and conditions"; three buttons: "Register", "Go to Courses", and "Login as admin".

Each admin has his own pin in it, admins' pins are: "8471", "2801", "9452"

```
admin.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        if (e.getButton() == MouseEvent.BUTTON1) {
            JFrame adminFrame = new JFrame("Admin");
            adminFrame.setSize(400, 400);
            adminFrame.setLayout(null);

            JLabel pinLabel = new JLabel("Enter your admin pin:");
            pinLabel.setBounds(20, 20, 200, 25);
            adminFrame.add(pinLabel);

            JTextField pinField = new JTextField();
            pinField.setBounds(20, 60, 160, 25);
            adminFrame.add(pinField);

            JButton submitPin = new JButton("Submit");
            submitPin.setBounds(20, 100, 100, 25);
            adminFrame.add(submitPin);

            submitPin.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    String pin = pinField.getText();
                    if (pin.equals("8471") || pin.equals("2801") || pin.equals("9452")) {
                        bubbleSort(students);

                        JFrame studentFrame = new JFrame("Students' Information");
                        studentFrame.setSize(400, 400);
                        studentFrame.setLayout(null);

                        int y = 20;
                        for (int i = 0; i < students.size(); i++) {
                            JLabel studentLabel = new JLabel(students.get(i).toString());
                            studentLabel.setBounds(20, y, 350, 25);
                            studentFrame.add(studentLabel);
                            y += 30;
                        }

                        studentFrame.setVisible(true);
                    } else {
                        JOptionPane.showMessageDialog(adminFrame, "Error: Incorrect PIN", "Error", JOptionPane.ERROR_MESSAGE);
                    }
                }
            });
        }
    }
});

adminFrame.setVisible(true);
```

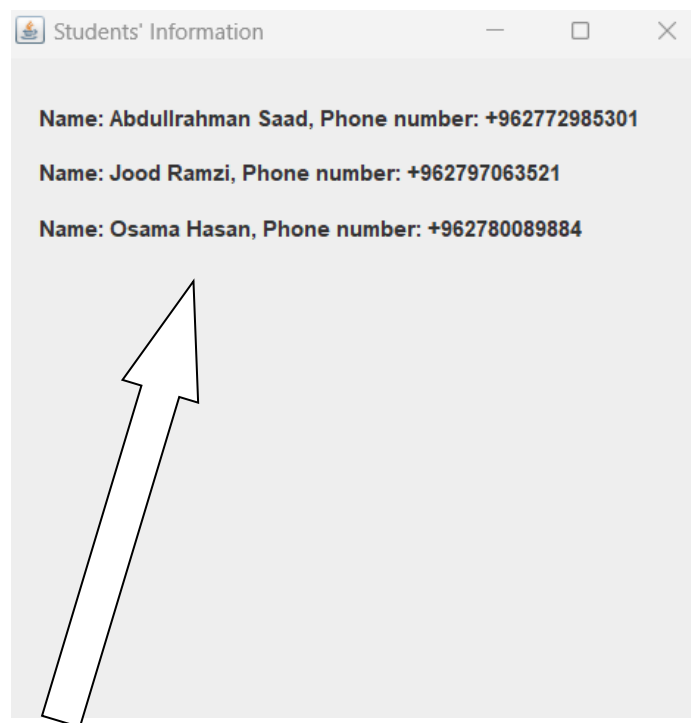
The screenshot shows a Java Swing window titled "Admin". It contains a label "Enter your admin pin:" and a text input field with the value "4927". Below the input field is a "Submit" button. An error dialog box is overlaid on top of the window, titled "Error", with a red 'X' icon and the message "Error: Incorrect PIN". The dialog has an "OK" button.

The screenshot shows the same "Admin" window. The text input field now contains the value "2801". The "Submit" button is still visible below it.

```
//Arrange the students' names in the alphabetical order
private static void bubbleSort(ArrayList<Student> students) {
    int n = students.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - 1 - i; j++) {
            if (students.get(j).compareTo(students.get(j + 1)) > 0) {
                // Swap students[j] and students[j+1]
                Student temp = students.get(j);
                students.set(j, students.get(j + 1));
                students.set(j + 1, temp);
            }
        }
    }
}
```

- I used bubble sort algorithm to order the names in the ArrayList alphabetically

After clicking submit in the admin frame:



- The program appears for the admin the student data (names and phone numbers), and the names are arranged in the alphabetical order, as it appears here even though the names were recorded in another order

7. To ensure good code quality, we must follow the following steps:

- Use a coding standard
- Analyze the code (automatically) – before code reviews; it's best to analyze code as soon as it's written.
- Enforce the rules – follow code review best practices & take advantage of automated tools
- Refactor legacy code (if necessary) – clean up your codebase and lower its complexity.

But remember that quality code is only part of the puzzle that makes up high-quality software. The quality of the code is essential, but not sufficient for creating high-quality software.

(Anon., 2024)

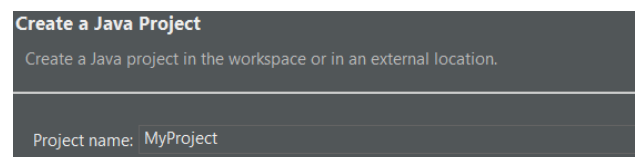
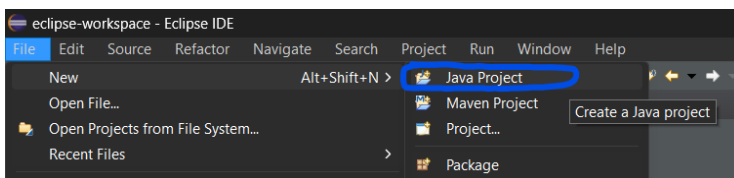
8. Testing Code Before Execution is very important for many factors, including:

- Ensuring the safety of the code from any errors, and repairing errors early, if any
- Ensure the quality of the code and that the expected outputs is what will actually appear in the desired way
- Building confidence, the trust between the developers and users is the most important thing for any company, and it must always be sure of everything the user will need to be sound to win his confidence

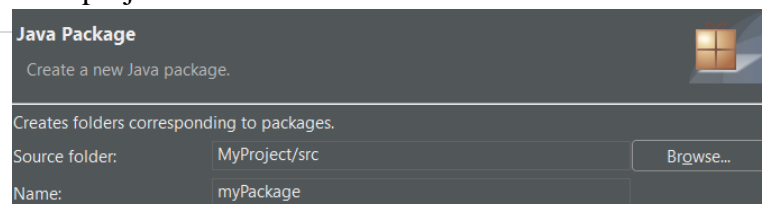
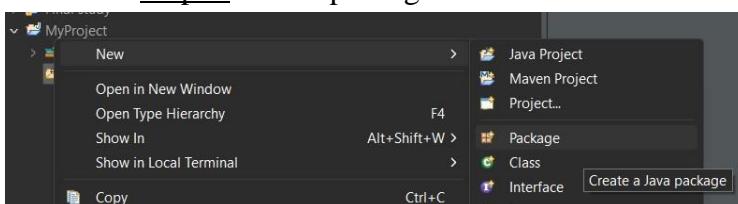
Part 3:

1.

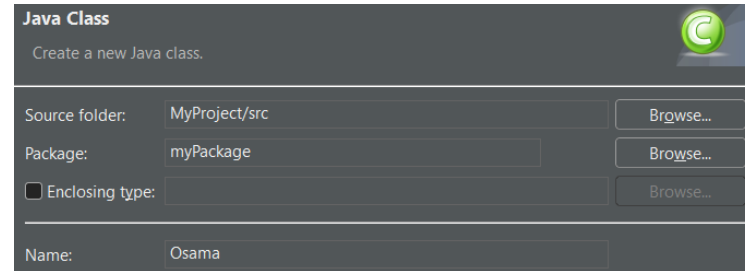
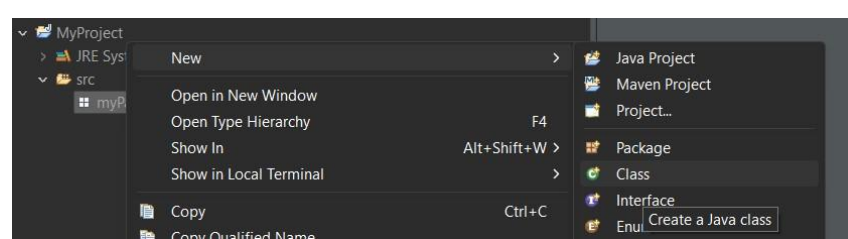
Step 1: Create a java project from the eclipse IDE menus and name it.



Step 2: Create packages inside the src folder within the project we created and name them.



Step 3: Create classes within each package and name them.



Step 4: Begin to work and coding through the paradigms we have learned.

Naming conventions should be applied during naming. We can create many packages and categories within the project and apply the principles we have learned to them. One project can be an application, a website, or a system for selling something.

2. The IDE features things that make it smoother and easier to code, including:

- Speed while writing and compiling code, through auto-completion and shortcuts in the IDE.
- It provides an organized structure for the classes within the same project and shows their connection to each other.
- Gives some hints to the programmer while writing the code, compiling, and other matters related to coding.

3. Debugging is one of the most important Features in the IDE that helps in discovering and solving errors easier by using debugging Tools, which represents features, such as:

- Breakpoints are a tool that help us discover errors through step execution by narrowing the scope of the error and allowing us to inspect fewer lines of code.
- Step Execution: Stepping into, over and out, helps us find the problem and check each line individually by stepping between them

- Exception Handling: It helps us to discover the root cause after the process of narrowing the error and by Step Execution, IDE can provide detailed information about the exception.

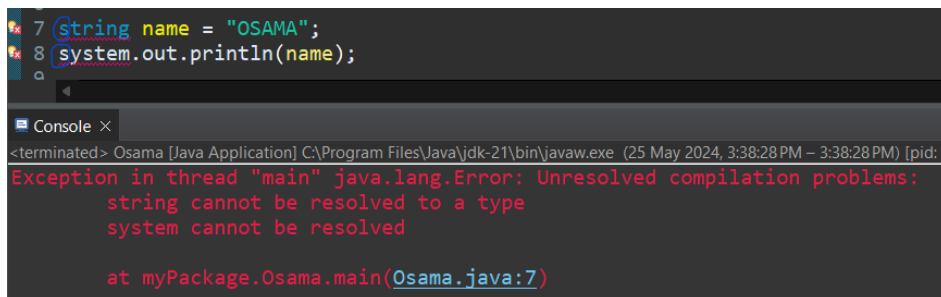
4. Using the IDE is much more efficient than not using it. The IDE provides an integrated and smooth work environment due to the clear and arranged features and menus. The coder can start and finish work comfortably, and this is what makes it easier, faster and much more efficient than not using the IDE. The presence of the auto-code completion, hints, error detection, etc., facilitates and speeds up the process of writing code and completion, makes the IDE more comfortable for writing code, and certainly makes it more efficient.

However, for a programmer who programs many things every day for long periods of time, using the IDE consumes a lot of space and resources, so in small and simple codes it is better not to use the IDE, to save as much space and resources as possible for larger projects.

Part 4:

1. The biggest problem the coder faces while writing any code is errors, so understanding the types of errors and determining the type of error that occurs to the writer while writing the code is very necessary to confront it. The types of programming errors are as follows:

1. Syntax error, which occurs while writing code when programming rules are violated. When its occur, the execution phase cannot be reached, ex:



```
7 string name = "OSAMA";
8 system.out.println(name);
```

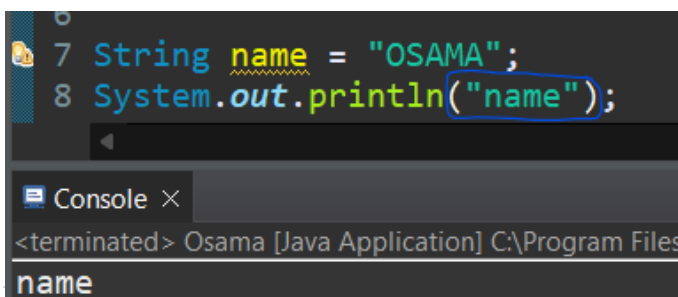
Console X

<terminated> Osama [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (25 May 2024, 3:38:28 PM - 3:38:28 PM) [pid: 2...]

Exception in thread "main" java.lang.Error: Unresolved compilation problems:
string cannot be resolved to a type
system cannot be resolved

at myPackage.Osama.main(Osama.java:7)

2. Logical Error, which occurs in the execution phase, In this case, the code is syntactically correct, but what is meant by the code is not the same as the output that appeared, ex:



```
7 String name = "OSAMA";
8 System.out.println("name");
```

Console X

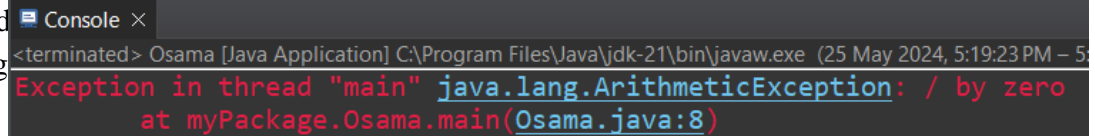
<terminated> Osama [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (25 May 2024, 3:38:28 PM - 3:38:28 PM) [pid: 2...]

name

3. Run time error, which occur as a crash after the execution phase, occurs when an exception occurs in the inputs, ex:

```
7 int num1 = 0;
8 int num2 = 5/num1;
9 System.out.println(num2);
```

Programming errors are the failure of the success of the code, due to errors produced. Understanding

A screenshot of a Java IDE's console window. The title bar reads 'Console x'. The text in the console shows a terminated Java application: '<terminated> Osama [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe (25 May 2024, 5:19:23 PM - 5)'. Below this, an exception is displayed in red and blue text: 'Exception in thread "main" java.lang.ArithmeticException: / by zero at myPackage.Osama.main(Osama.java:8)'.

2.

Step 1: Gather information about the error, The data must be examined after collecting it to ensure that nothing is forgotten and verifying its validity.

Step 2: Isolate the Error, by putting Breakpoints.

Step 3: Identify the Error, by searching and narrowing the range by Stepping over and return.

Step 4: Determine how to fix the error.

Step 5: Apply and Test, to make sure that the error has been fixed 100%.

(Nduta, 2023)

3. Breakpoints are a debugging tool that allows a program to be temporarily stopped at a specific point. This helps identify the location of logical or runtime errors that cannot be automatically determined by the IDE. By examining variable values, testing each line, and stepping through the code, developers can identify where the error occurs and understand the code's behavior.

4. The debugging process is very important to verify the integrity of the code by having the developers examine the code and identify the problem occurring early, and this is what maintains the security of the application. Here are examples of security vulnerabilities that debugging helps identify and address:

- Cross Site Scripting
- SQL Injection
- LDAP Injection
- Cross Site Request Forgery
- Insecure Cryptographic Storage

(Anon., 2012)

5.

```
submitPin.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String pin = pinField.getText();
        if (pin.equals("8471") || pin.equals("2801") || pin.equals("9452")) {
            bubbleSort(students);

        }
        JFrame studentFrame = new JFrame("Students' Information");
        studentFrame.setSize(400, 400);
        studentFrame.setLayout(null);

        int y = 20;
        for (int i = 0; i < students.size(); i++) {
            JLabel studentLabel = new JLabel(students.get(i).toString());
            studentLabel.setBounds(20, y, 350, 25);
            studentFrame.add(studentLabel);
            y += 30;
        }

        else {
            JOptionPane.showMessageDialog(adminFrame, "Error: Incorrect PIN", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
});
```

Here, I had a logical error, and I did not know its cause. I put everything I needed, and the code did not work correctly. I put a breakpoint and made sure that the problem lay here. During step execution, I discovered that the location of the curly bracket was wrong and that the if statement's scope was empty in this case, so I corrected it to this:

```
submitPin.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String pin = pinField.getText();
        if (pin.equals("8471") || pin.equals("2801") || pin.equals("9452")) {
            bubbleSort(students);

            JFrame studentFrame = new JFrame("Students' Information");
            studentFrame.setSize(400, 400);
            studentFrame.setLayout(null);

            int y = 20;
            for (int i = 0; i < students.size(); i++) {
                JLabel studentLabel = new JLabel(students.get(i).toString());
                studentLabel.setBounds(20, y, 350, 25);
                studentFrame.add(studentLabel);
                y += 30;
            }

            studentFrame.setVisible(true);
        } else {
            JOptionPane.showMessageDialog(adminFrame, "Error: Incorrect PIN", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
});
```

Debugging is a feature provided by the IDE to make it easier for the coder to discover the error and correct it, so that the coding process becomes more enjoyable and not psychologically tiring due to the coder spending the entire time searching for the error. I benefited from it a lot and I will benefit from it even more.

Part 5:

1. You can write a complex and not simple code and it may work well for you, but it will not be easy for others to understand or read or develop in the future. Therefore, the code working well does not mean that it has become perfect. Rather, there are other important things other than just writing the code and running it, such as:

- Adding comments, to make it easier for others to understand the code and read it correctly, we should add some comments that explain the parameters and other things related to the code, whether they are single-line or multi-line comments.
- Indentation, to know this part belongs to which scope.
- The location of the opening and closing brackets, so that there is no confusion for the code reader about the beginning and end of loops and scopes in general.
- Braces are used around all statements.
- One declaration line, not several on the same line, to not confuse the code reader.
- One statement per line, not several on the same line, to not confuse the code reader.
- Choose meaningful and clear names and apply naming conventions.

2. There are many naming conventions in the world of coding, which help to understand the meaning of classes, parameters, etc., where the name has a clear meaning, such as:

1. Pascal case: The first letter of each word should be capital case, we use it usually for naming methods and classes, ex:

```
public class ProgramOne {  
}
```

2. Camel case: The first letter of the first word is lowercase, but the first letter of the rest words is uppercase, we use it usually for naming variables and method arguments, ex:

```
int cardNumber;
```

3. snake case: All letters of words are uppercase, and words are separated by an underscore, we use it usually for naming constants defined by the final keyword, ex:

```
static final int MIN_SALARY = 1000;
```

3. The most important reason for the success of any team or business is cooperation and the harmonious distribution of roles, and in the field of coding this becomes possible with the coding standards, the most important reasons that lead to achieving them and the team's success due to coding standards:

1. It enhances trust among team members, because everyone can understand the other's work because the code structure is clear to everyone, and anyone can check the other's task and make sure that he has not forgotten anything and can modify it with ease.
2. Reducing errors, by following coding standards. The steps are clear to the code writer, and this is what makes him not lost or forget anything while writing the code and makes identifying errors, whether from the code writer or the auditor, much easier. This makes the success of the project greater and more feasible.
3. It makes the work consistent and the style is uniform among everyone. It appears as if one person performed all the work, and this enhances the quality and efficiency of the code.

References:

tutorchase. (2024). *How does one ensure that an algorithm is both effective and efficient?* / *TutorChase*. [online] Available at: <https://www.tutorchase.com/answers/ib/computer-science/how-does-one-ensure-that-an-algorithm-is-both-effective-and-efficient>. (Accessed: 29 May. 2024).

Frame of Essence (2017). How do computers read code? [online] YouTube. Available at: <https://www.youtube.com/watch?v=QXjU9qTsYCc&t=508s> [Accessed 29 May. 2024].

EncoraJobs. (2024). Improving Code Quality: steps and best practices. [online] Available at: <https://careers.encora.com/talent-hub/encora-best-practices-and-steps-quality-code-improving#:~:text=The%20best%20way%20to%20guarantee> [Accessed 1 Jun. 2024].

careerfoundry. (2023). What is Debugging? A Simple Guide for Beginners. [online] Available at: <https://careerfoundry.com/en/blog/web-development/what-is-debugging/>. [Accessed 8 Jun. 2024].

Veracode. (2012). Application Security Vulnerability: Code Flaws, Insecure Code. [online] Available at: <https://www.veracode.com/security/application-security-vulnerability-code-flaws-insecure-code>. [Accessed 9 Jun. 2024].

Any question for which I did not include a reference, this means that the answer is from my knowledge after studying the slides and videos of the course.