GOVERNMENT SHAHID AKBAR ALI SCIENCE & TECHNOLOGY COLLEGE

BALIADANGI, THAKURGAON

AFFILIATED BY

HAJEE MOHAMMAD DANESH SCIENCE & TECHNOLOGY UNIVERSITY, DINAJPUR

PROJECT NAME: SPACE INVADERS

Course Title: Object Oriented Programming(Java) Sessional

Department: Computer Science & Engineering

Level -2 Semester- II

Course Code: CSE 254

Presented By:

Partha Sarathi Roy	Md.Alamgir Hossain	Shahed Khan
Student ID: 2120104	Student ID: 2120108	StudenID:1920105
Session: 2021	Session: 2021	Session: 2019

ABSTRACT

The **Space Invaders** project is a Java-based implementation of the classic arcade game where players control a spaceship to defend against descending aliens. The game incorporates graphics, sound, and user interactions through keyboard inputs, offering an engaging gameplay experience. The main features include the ability to move the spaceship, shoot bullets, and collide with aliens, along with scoring mechanisms and a game-over state. The project leverages object-oriented programming principles, utilizing classes and data structures to represent game elements and manage game logic effectively. The implementation highlights the importance of graphics rendering, event handling, and collision detection in game development.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Space Invaders is one of the most iconic video games, initially developed in 1978 by Tomohiro Nishikado.[5] The game involves controlling a spaceship that moves horizontally at the bottom of the screen while aliens move towards the player from the top. The player must shoot the aliens while avoiding their attacks. The game's difficulty increases as the aliens descend closer and faster with each level.

This project aims to recreate this classic game using Java, focusing on object-oriented design and interactive gameplay. The game offers an opportunity to learn game design principles while implementing advanced programming concepts.

1.2 PURPOSE OF THE PROJECT

The purpose of this task is to recreate the classic Space Invaders game the usage of Java, exploring fundamental game improvement standards such as animation, user input, and collision detection.

1.3 OBJECTIVES

The objectives of this project are:

- To create an interactive 2D game using Java and its Swing package.
- To implement a game loop that handles user input and collision detection in real-time.
- To simulate enemy movement and increase difficulty over time.
- To display scores and handle end-game scenarios.
- To ensure the project follows clean code practices, such as modular design and reusable components.

1.4 SCOPE

The game will consist of:

- Developing a 2D game interface using Java Swing.
- Implementing essential game mechanics like movement, shooting, and collision detection.

- Managing multiple enemies (aliens) and projectiles (bullets) in real-time.
- Scoring system based on the number of aliens destroyed.
- Providing an interactive user interface where the player can restart the game after it ends.

1.5 LIMITATIONS

- The project is limited to a single-player mode.
- The enemy's attack patterns are simplistic and do not involve advanced AI behaviours.
- The game does not support sound effects or background music due to time constraints.
- The game only runs in a fixed window size and does not adapt to various screen sizes or resolutions.

CHAPTER 2
GAME DESIGN

2.1 GAME MECHANICS

The core mechanics of the Space Invaders game involve controlling a ship that moves horizontally at the bottom of the screen. The primary objective is to shoot down waves of descending aliens using bullets. Players can move the ship left or right and fire bullets by pressing the space bar. Key mechanics include:

- **Movement:** The ship moves left or right across the screen boundaries. The alien enemies also move horizontally, changing direction when they hit the edge of the screen.
- **Shooting:** Players can shoot bullets upwards, which will move towards the aliens. Each bullet is a small rectangle that disappears when it goes off-screen or hits an alien.
- **Collisions:** The game includes collision detection between bullets and aliens, allowing players to destroy aliens and gain points.
- **Game Over:** The game ends when any alien reaches the bottom of the screen, signalling a loss for the player.

2.2 GAME ELEMENTS

The game consists of several key elements:

- **Player Ship:** The player's ship is represented as a graphic that the player controls. It can be moved left and right to dodge incoming aliens.
- **Aliens:** Different types of aliens are displayed, each with its unique appearance. They move in formations and become increasingly aggressive as the game progresses.
- **Bullets:** Bullets are shot from the player’s ship and are essential for defeating aliens.
- **Score:** A scoring system keeps track of points earned by shooting aliens, displayed at the top of the screen.

2.3 FLOW CHART

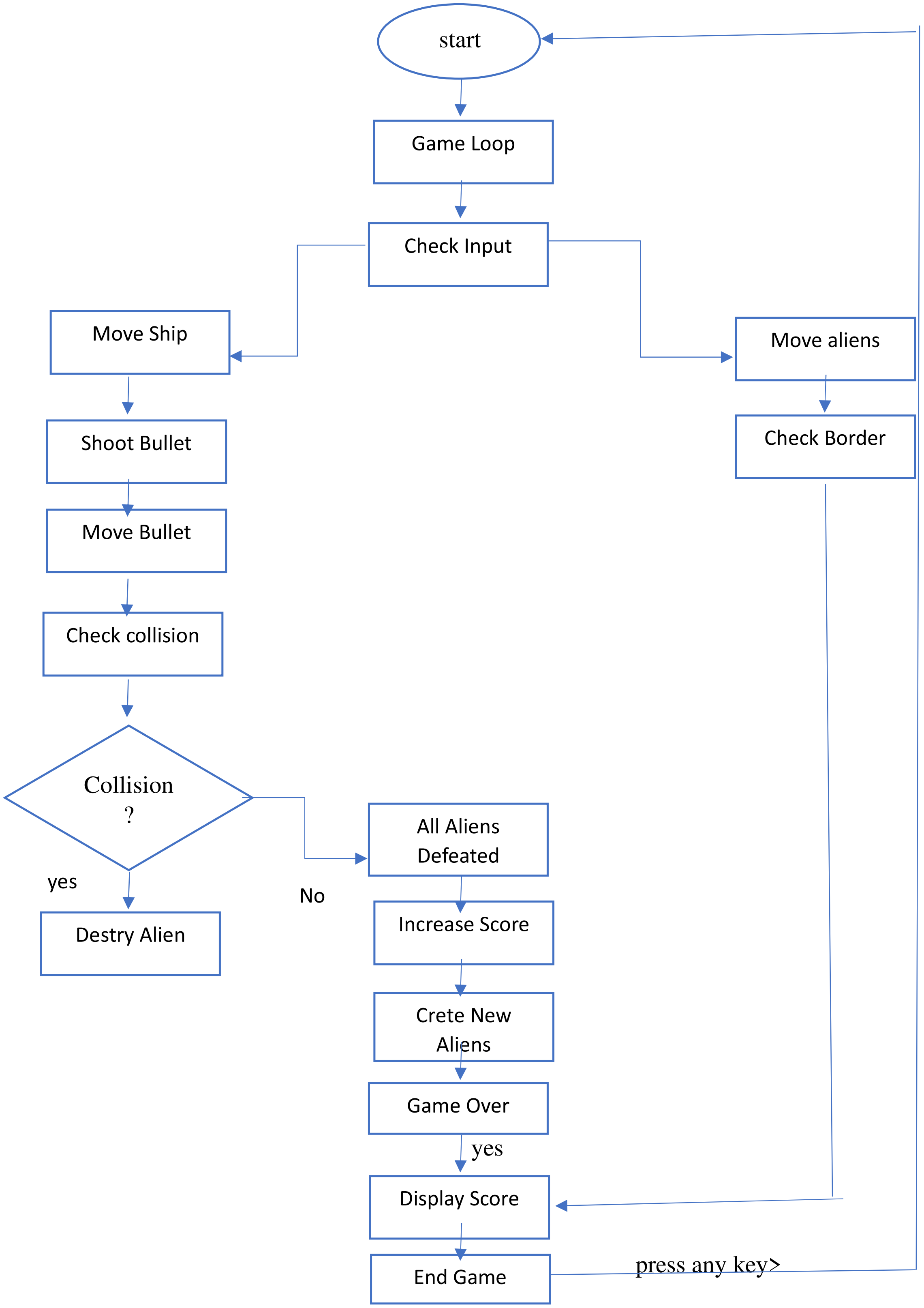


Fig 2.3: Flow Chart

2.4 USER INTERFACE

The user interface is designed to be intuitive and straightforward:

- **Game Screen:** The game displays a blue background where all action takes place. The player's ship, aliens, and bullets are drawn on this canvas.
- **Score Display:** The score is prominently displayed at the top left of the screen, updating in real-time as players defeat aliens.
- **Game Over Message:** When the game ends, a "Game Over" message appears on the screen, along with the player's final score.

CHAPTER 3
IMPLEMENTATION

3.1 PROGRAMMING LANGUAGE AND TOOLS

The **Space Invaders** game is implemented in **Java**, utilizing the **Swing** library for graphical user interface (GUI) components and event handling. The key tools and libraries used in the development include:

- **Java Development Kit (JDK)**: The core programming language used to write and compile the code.[1]
- **Swing**: A part of Java Foundation Classes (JFC) used for creating window-based applications, including the game panel and rendering graphics.[2]
- **Java AWT**: Used for handling graphics and event listeners.[3]

3.2 CODE STRUCTURE

The code is structured into a single primary class, SpaceInvaders, which extends JPanel and implements ActionListener and KeyListener. The main components of the code include:

- **Block Class**: A nested class representing game objects like the player ship, aliens, and bullets, encapsulating their properties (position, size, image) and states (alive, used).[1]
- **Game Variables**: Variables for managing game state, including the board dimensions, ship and alien properties, score, and game status.[1]
- **Initialization**: The constructor initializes game components, loads images, sets up the game loop, and creates aliens.
- **Game Loop**: Managed by a Timer, this loop calls the move() method to update game state and the repaint() method to render graphics at a consistent frame rate.

3.3 GAME LOGIC

The game logic includes:

- **Movement**: The move() method controls the movement of aliens and bullets. Aliens move horizontally and descend upon hitting the screen edges. Bullets move upward and check for collisions with aliens.

- **Collision Detection**: The detect Collision() method checks for collisions between bullets and aliens, updating their states and scores accordingly.
- **Game Over Condition**: If any alien reaches the player’s ship, the game ends, and a "Game Over" message is displayed.

3.4 GRAPHICS AND ANIMATION

Graphics are rendered in the paint Component(Graphics g) method, which uses:

- **Drawing Images**: The player ship, aliens, and bullets are drawn on the game panel using their respective images.
- **Background Colour**: The background of the game panel is set to blue for a space-like appearance.
- **Score Display**: The current score is displayed on the screen using a white font, and a game over message is shown if the player loses.

3.5 INPUT HANDLING

User inputs are handled through the Key Listener interface:

- **Key Press Events**: The key Released(Key Event e) method manages player controls, allowing the ship to move left and right using the arrow keys and fire bullets using the space bar.
- **Game Restart**: If the game is over, pressing any key resets the game state, allowing the player to start a new game.

CHAPTER 4

PROJECT OUTPUT

When the project runs, the following outputs and features are expected:

4.1 GAME INTERFACE

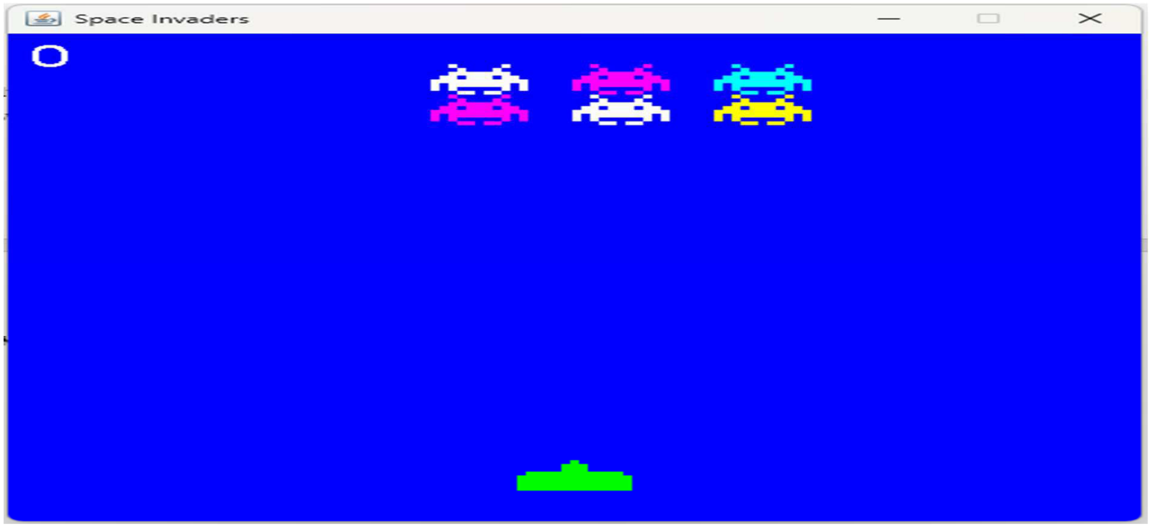


Fig 4.1: Game Interface

A 512x512 black game board is displayed .The player controls a spaceship (displayed as an image) at the bottom of the screen.

4.2 ALIENS

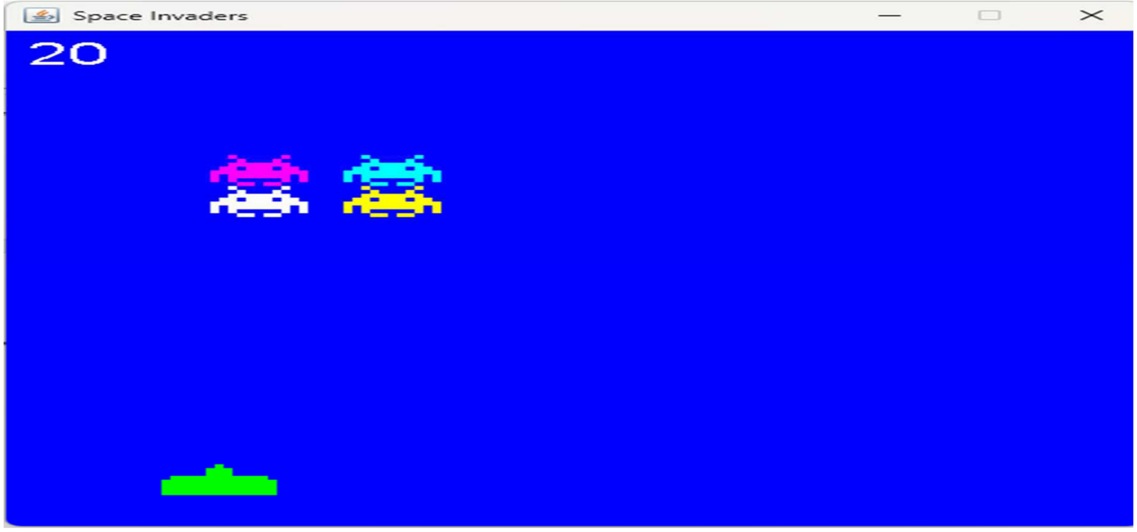


Fig 4.2: Aliens

Multiple alien images appear at the top of the screen, arranged in rows and columns. Aliens move horizontally across the screen and descend towards the player's ship.

4.3 PLAYER INTERACTION

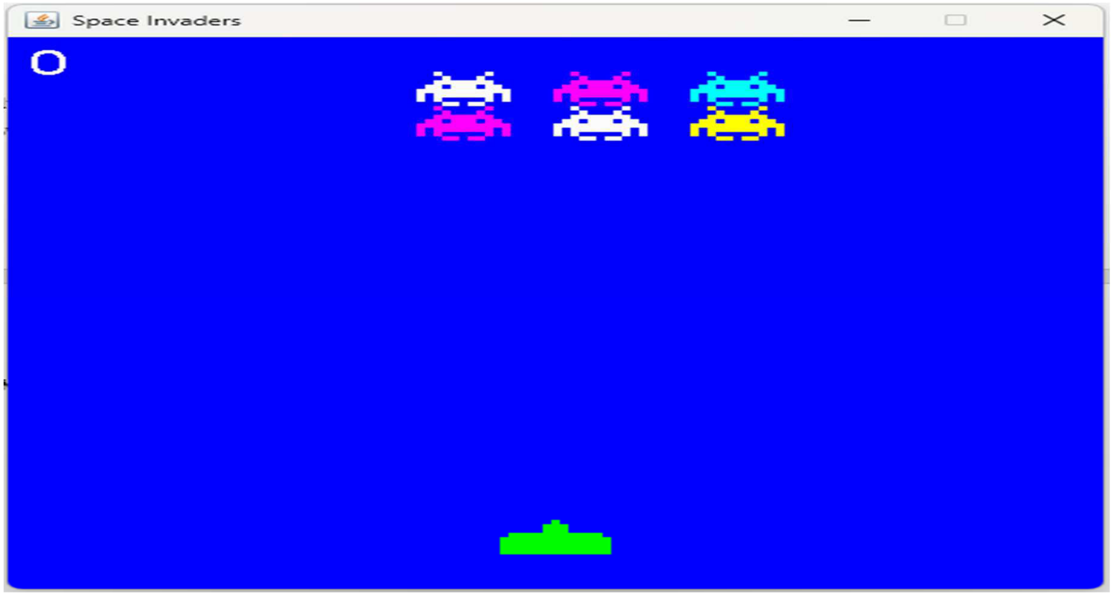


Fig 4.3: Player Interaction

The player can move the spaceship left and right using arrow keys .Pressing the spacebar will shoot bullets from the spaceship.

4.4 SCORING SYSTEM

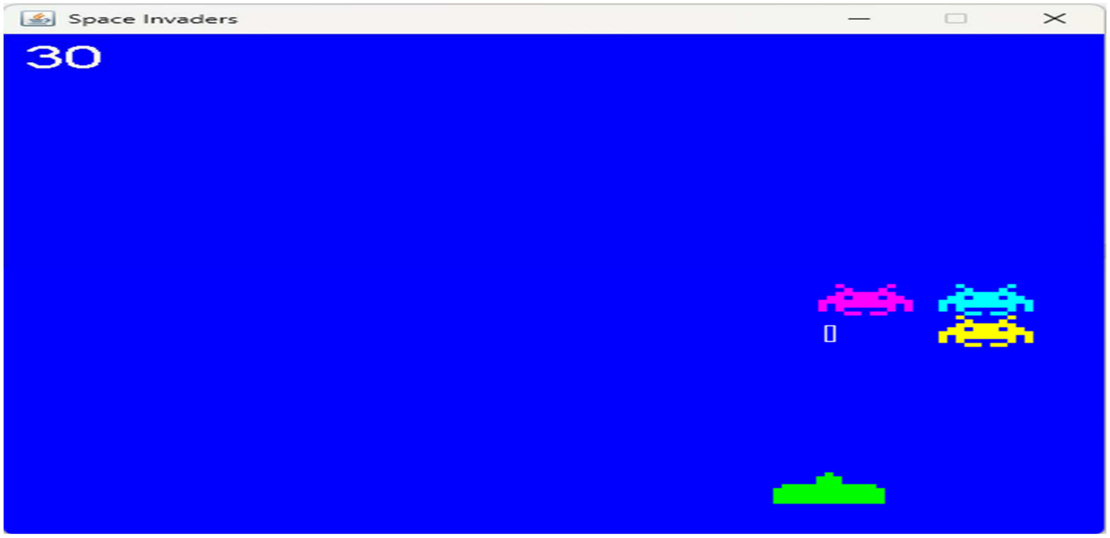
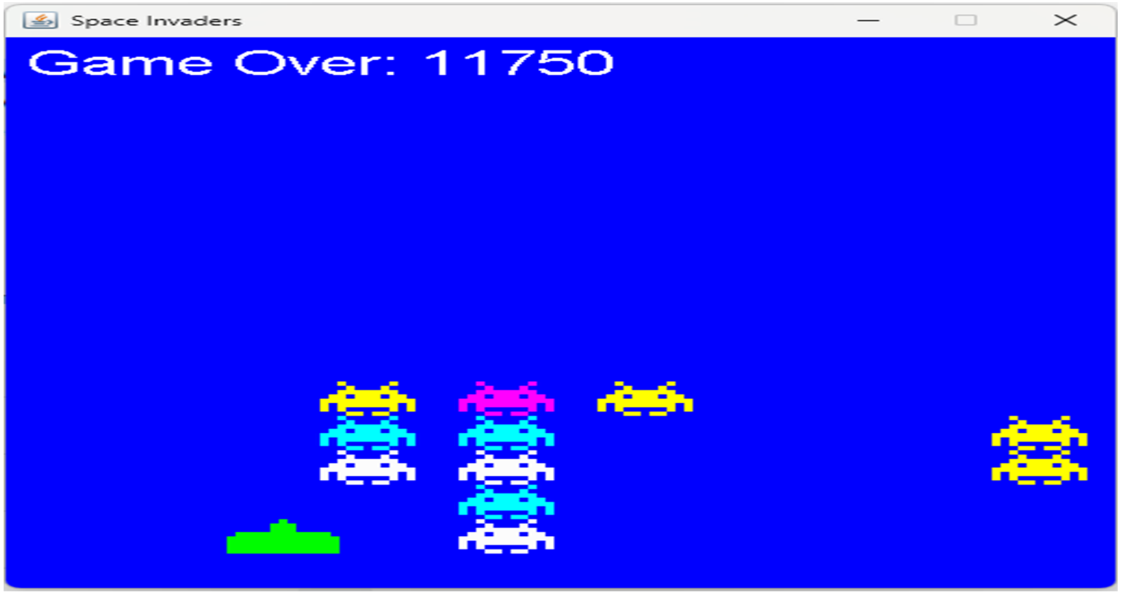


Fig 4.4: Scoring System

The score increases by 10 points for each alien shot down. The score is displayed at the top of the screen.

4.5 GAME OVER



Game Over: 11750

Fig 4.5: Game Over

If an alien reaches the bottom of the screen, the game ends, and a "Game Over" message is displayed.

4.6 RESTART OPTION

No need any Images

Pressing any key after a game over will reset the game state, allowing the player to play again.

CHAPTER 5

CONCLUSION

This Space Invaders project effectively demonstrates the use of object-oriented programming and basic game development techniques in Java. It features interactive gameplay with alien invaders, a player-controlled ship, and dynamic bullet mechanics. The game uses a timer for real-time movements and events, while collisions are handled efficiently. The simplicity of the design allows for easy scalability, such as adding more aliens or increasing difficulty. Overall, this project showcases fundamental game design concepts and provides a foundation for more complex game development projects.

FUTURE IMPROVEMENTS

While the current implementation of the Space Invaders project is functional, several improvements can be made to enhance the gameplay experience and code quality:

- Enhanced Graphics and Sound
- Scoring and Leaderboard System
- Level Design and Difficulty Scaling
- Power-Ups and Bonuses
- Responsive Design
- Testing and Bug Fixing

REFERENCES

[1] Java: A Beginner's Guide by **Herbert Schildt**

[2] Java Swing: A Tutorial by **Cay Horstmann and Gary Cornell**

[3] Oracle Java Documentation: https://docs.oracle.com/en/java/

[4] Tutorials Point Java Swing Tutorial: https://www.tutorialspoint.com/swing/index.htm

[5] Overview of point-of-sale systems: https://en.wikipedia.org/wiki/Point_of_sale

[6] Stack Overflow: https://stackoverflow.com/