ICT167 Principle of Computer Science

Table of Contents

[**1.** **Title** 3](#_Toc162530113)

[**2. Requirements/Specification** 4](#_Toc162530114)

[**3. User Guide** 5](#_Toc162530115)

[**4. Structure/Design/Algorithm** 5](#_Toc162530116)

[**5. Limitations** 5](#_Toc162530117)

[**6. Testing** 5](#_Toc162530118)

# **1. Title**

**Student Management System Documentation:** Authored by Keith Neo Kai Si on March 28, 2024, the Student Management System Documentation provides an overview of the functionalities of the Java-Based Student Management System. The system is designed to handle student data through CSV files and comprises several Java files, including **CsvHandler.java**, **Client.java**, **Student.java**, **Student\_Research.java**, **Student\_Course.java**, **Unit.java**, **Unit\_Course.java**, and **Research.java**. Its core objectives include efficiently loading student information from **student.csv**, adding marks from **student\_marks.csv**, student removal, information display, and exporting organized student data to a CSV file. Employing Java’s object-oriented programming principles, such as inheritance and polymorphism, the system effectively models various student types and their enrolment in different units. Furthermore, the documentation elaborates on Java’s file-handling capabilities, illustrating the system’s CSV file reading and writing processes for sustained data integrity. This comprehensive guide serves as a valuable reference for developers seeking to implement or enhance student management systems using Java.

# **2. Requirements/Specification**

The Student Management System is created to effectively handle and manage student information, such as importing student details from CSV files, adding marks to students, deleting students, displaying student information, and exporting organized student data back to a CSV file. The system assumes that the input CSV files are correctly formatted, with student.csv containing student type, name, and ID and student\_marks.csv containing student IDs followed by marks or course and marks. The desired output is a sorted list of students with their marks included, which can be saved to a new CSV file.

The system’s design utilizes fundamental object-oriented programming concepts:

**1. Inheritance**: The Student class acts as the base class, with Student\_Research and Student\_Course extending it to represent different types of students. This allows for reusing code and adding specific attributes and methods for research and course students.

**2. Polymorphism**: The reportGrade method in both Student\_Research and Student\_Course classes overrides the reportGrade method in the Student class, showcasing polymorphism. This enables different implementations of the method based on the student type.

**3. Dynamic Binding**: The system employs dynamic binding to determine the appropriate method to call at runtime. For instance, when calling reportGrade on a student object, the JVM decides whether to call the Student, Student\_Research, or Student\_Course implementation based on the actual type of the student object.

**4. Sorting Algorithm**: The system utilizes an insertion sort algorithm to arrange students by their IDs. This is done using the sortStudents method, which sorts the list of students in ascending order of their IDs.

**5. Handling CSV Files**: The system uses Java’s built-in Scanner class to read from CSV files. It reads each line of the student\_marks.csv file, separates the line into components based on the comma separator, and processes the data accordingly. For writing to a CSV file, the system uses PrintWriter to output the sorted list of students to a new file.

This documentation presents a comprehensive overview of the requirements, specifications, and implementation details of the Student Management System, ensuring that developers and users comprehend the system’s functionality and how it fulfills the specified needs.

# **3. User Guide**

# **4. Structure/Design/Algorithm**

# **5. Limitations**

# **6. Testing**