Java Basics - Constructor, Array, and Function Overloading

Objective: Write a Java program that demonstrates the use of constructors, arrays, and function overloading.

Requirements:

- Create a new Java class that represents a student with attributes such as name and age.
- Use different types of constructors to initialize the student attributes.
- Create an array to store multiple student objects.
- Implement function overloading to display student information.

Exercises:

- Write a Java program that calculates the average age of students in the array.
- Write a Java program that sorts the students by name.
- Write a Java program that demonstrates the use of a constructor with default values for attributes.

Hint: Here is an example code snippet to get you started:
Constructor:

```
// Student.java
public class Student {
    private String name;
    private int age;

    // Default constructor
    public Student() {
        this.name = "Unknown";
        this.age = 0;
    }

    // Parameterized constructor
    public Student(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Copy constructor
    public Student(Student student) {
        this.name = student.name;
        this.age = student.age;
    }

    // Getter methods
```

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

    public int getAge() {
        return age;
    }
}
```

Array:

```
// StudentArray.java
public class StudentArray {
    public static void main(String[] args) {
        // Create an array to store student objects
        Student[] students = new Student[5];

        // Initialize student objects and add them to the array
        students[0] = new Student("Alice", 20);
        students[1] = new Student("Bob", 22);
        students[2] = new Student("Charlie", 21);
        students[3] = new Student(); // default constructor
        students[4] = new Student(students[0]); // copy constructor

        // Display student information
        for (int i = 0; i < students.length; i++) {
            System.out.println("Name: " + students[i].getName() + ",
            Age: " + students[i].getAge());
        }
    }
}
```

Function Overloading:

```
// DisplayStudentInfo.java
public class DisplayStudentInfo {
    // Function to display student information by name
    public void display(String student_name) {
        System.out.println("Student Name: " + student_name);
    }

    // Function to display student information by name and age
    public void display(Student student_ref) {
        System.out.println("Student Name: " + student_ref.getName() + ",
        Age: " + student_ref.getAge());
    }

    public static void main(String[] args) {
```

```
        DisplayStudentInfo displayInfo = new DisplayStudentInfo();

        // Display information using function overloading
        displayInfo.display("Alice");
        displayInfo.display(new Student("Bob", 22));
    }
}
```

Bonus Exercises:

- Write a Java program that implements a method to remove a student from the array based on their name.

  Hint: Use a loop to iterate through the array and find the student with the matching name, then shift the remaining students to fill the gap.

- Write a Java program that implements a method to update a student's age in the array based on their name.

  Hint: Use a loop to iterate through the array and find the student with the matching name, then update their age.

- Write a Java program that calculates the total number of students in the array who are above a certain age (e.g., 21).

  Hint: Use a loop to iterate through the array and count the students who meet the condition.

- Write a Java program that sorts the students in the array based on their age in descending order.

  Hint: Use a sorting algorithm such as bubble sort or selection sort.

- Write a Java program that implements a method to search for a student in the array based on their name and returns their age.

  Hint: Use a loop to iterate through the array and find the student with the matching name, then return their age.

- Write a Java program that implements a method to display the student information in a tabular format (e.g., using a table or a grid).

  Hint: Use a loop to iterate through the array and print the student information in a tabular format.

- Write a Java program that implements a method to save the student information in a file and read it back from the file.

  Hint: Use the FileWriter and FileReader classes to write and read the student information to and from a file.