EcoPoints Recycling Tracker - Guided Project

Estimated Duration: 60 minutes

Learning objectives

After completing this project, you will be able to demonstrate the following skills:

- Apply object-oriented programming to design Household and RecyclingEvent classes using encapsulation.
- Construct and manage ArrayList and HashMap collections to store multiple households and their recycling events.
- Use Java's Date and Time API to track household join dates and recycling dates.
- Implement console-based input handling to register households and log recycling activity.
- Apply conditional logic and loops to process user choices and navigate program options.
- Handle invalid inputs and file-related errors using Java's exception handling.
- Use Java File I/O and serialization to persist and reload program data.
- Generate reports that summarize total recycled weight and eco points across households.

You'll complete this project with the ability to comfortably work with:

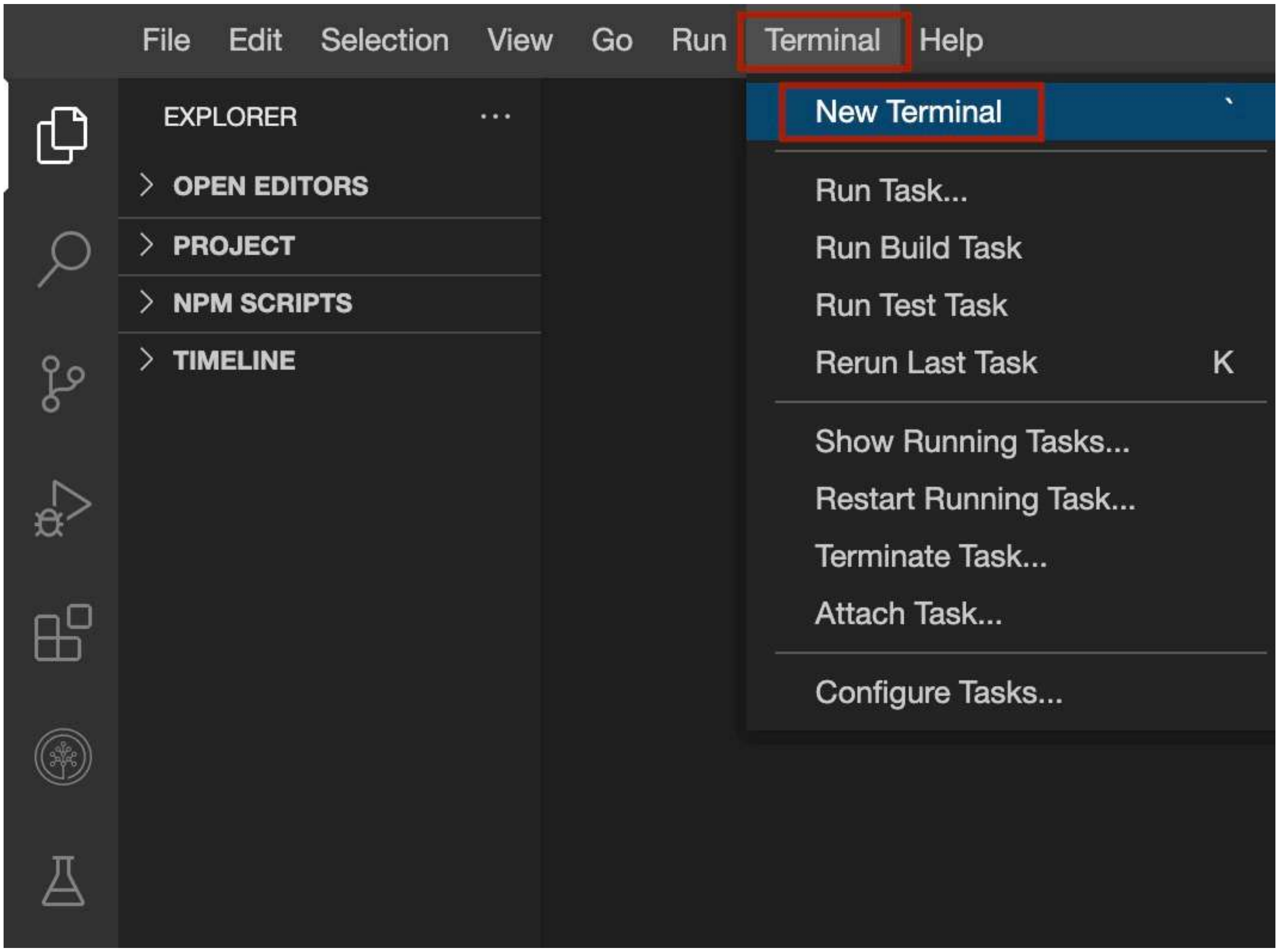
- Basics of Java programming
- Strings and string operations
- Operators and Data Types
- Exceptions
- `for` loops and the `while` Loop
- Conditional statements
- Arrays, Sets, and Maps
- Basic methods and functions
- Object-Oriented Programming Concepts
  - Encapsulation
  - Overriding
  - Polymorphism

You will complete this project using the Cloud IDE with Java preinstalled. You can complete this lab in your own IDE if you pre-install Java. This code is compatible for the IDE using JDK version 21.

All the code provided has been tested in the Cloud IDE. This lab includes full starter code, clear instructions, and lots of practice opportunities for you to apply the core Java skills you have learned so far. Please take advantage of these opportunities to understand how the provided code works.

Initial setup

1. Open a new terminal.



2. Create a directory named `EcoPointsRecyclingTracker` under `/home/project` by running the following command in the terminal window.

```
mkdir EcoPointsRecyclingTracker
```

3. Change to the project directory.

```
cd EcoPointsRecyclingTracker
```

The object blueprint classes

1. Create a new file named `RecyclingEvent.java`. You will create the `RecyclingEvent` class first, as it will be a member of the `Household` class. When it's time to define the `Household` class, `RecyclingEvent` should already be defined.

   ```
   touch RecyclingEvent.java
   ```

2. Open the `RecyclingEvent.java` file and paste the following content in the file to create the class.

Open **RecyclingEvent.java** in IDE

```
//Task 1
import java.io.Serializable;
import java.time.LocalDate;
/**
 * Represents a single recycling event for a household.
 */
public class RecyclingEvent implements Serializable {
    private String materialType;
    private double weight; // in kilograms
    private LocalDate date;
    private double ecoPoints;
    public RecyclingEvent(String materialType, double weight) {
        this.materialType = materialType;
        this.weight = weight;
        this.date = LocalDate.now();
        this.ecoPoints = weight * 10; // 10 points per kg
    }
    public String getMaterialType() {
        return materialType;
    }
    public double getWeight() {
        return weight;
    }
    public LocalDate getDate() {
        return date;
    }
    public double getEcoPoints() {
        return ecoPoints;
    }
    @Override
    public String toString() {
        return "Date: " + this.date +
                "\nMaterial Type: " + this.materialType +
                "\nWeight: " + this.weight +
                "\nEcopoints: " + this.ecoPoints;
    }
}
```

- `import java.io.Serializable;` - This lets the class be serializable, meaning you can save this object to a file and load it back later. No recycling event gets left behind.
- `import java.time.LocalDate;` - Brings in LocalDate to store the date when the recycling happened.
- `public RecyclingEvent(String materialType, double weight)` - This is the constructor. Whenever you make a new RecyclingEvent, you give it materialType and weight. The constructor does the math. It fills in the date as today and calculates eco points (10 points per kg).
- `public String toString` - Overrides the toString method in Object class to provide the stringified version of the Recycling event class.

3. Create a new file named `Household.java`.

```
touch Household.java
```

4. Open the file and paste the following content in it to create the class.

Open **Household.java** in IDE

```
//Task 1
import java.io.Serializable;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
/**
 * Represents a household participating in the Eco-Points program.
 */
public class Household implements Serializable {
    private String id;
    private String name;
    private String address;
    private LocalDate joinDate;
    private List<RecyclingEvent> events;
    private double totalPoints;
    public Household(String id, String name, String address) {
        this.id = id;
        this.name = name;
        this.address = address;
        this.joinDate = LocalDate.now();
        this.events = new ArrayList<>(); //Task 2
        this.totalPoints = 0.0;
    }
    public String getId() { return id; }
    public String getName() { return name; }
    public String getAddress() { return address; }
    public LocalDate getJoinDate() { return joinDate; }
    public List<RecyclingEvent> getEvents() { return events; }
    public double getTotalPoints() { return totalPoints; }
    public void addEvent(RecyclingEvent event) {
        this.events.add(event);
        this.totalPoints += event.getEcoPoints();
    }
    public double getTotalWeight() {
        double total = 0.0;
        for (RecyclingEvent event : events) {
            total += event.getWeight();
        }
        return total;
    }
}
```

- `import java.io.Serializable;` - Lets you save and load the object to and from a file so no data gets lost.
- `import java.time.LocalDate;` - Includes the modern date class for storing the date when the household joined.
- `import java.util.ArrayList` and `import java.util.List` - Use these lists to store multiple recycling events.
- `public Household(String id, String name, String address)` - This code is the constructor. When you register a new Household, a new Household object is created. You specify the household's ID, name, and address. The joinDate is set to the current date. The recyclingEvents collection is initialized as an empty list and the points are initialized to 0.

The main application

1. Create a new file named `EcoPointsRecyclingTracker.java`. This file is where your console application is going to take shape.

```
touch EcoPointsRecyclingTracker.java
```

2. Open the `EcoPointsRecyclingTracker.java` file and paste the following content provided in the following steps into the file to create the class.

Open **EcoPointsRecyclingTracker.java** in IDE

3. Add the following code in the file. First, you will add the imports.

```
import java.io.*;
import java.util.*;
import java.time.LocalDate;
```

4. Now add the empty class structure inside which you will be adding the code for the application in the main method.

```
/**
 * Main app to run the Eco-Points Recycling Tracker.
 */
public class EcoPointsRecyclingTracker {
    //This is where the rest of the code will be added
}
```

5. Because this application is console-based, you need to instantiate the Scanner class to read from console. Paste following code inside the class definition.

```
    private static Scanner scanner = new Scanner(System.in);
```

6. You will then create a HashMap to collect all the Household objects. The Household objects must be mapped against their respective ID. When the application starts, the collection will be empty.

   ```
   private static Map<String, Household> households = new HashMap<>(); // Task 2
   ```

7. Add the main method for the class.

   ```
       public static void main(String[] args) {
           boolean running = true;
           while (running) {
               System.out.println("\n=== Eco-Points Recycling Tracker ===");
               System.out.println("1. Register Household");
               System.out.println("2. Log Recycling Event");
               System.out.println("3. Display Households");
               System.out.println("4. Display Household Recycling Events");
               System.out.println("5. Generate Reports");
               System.out.println("6. Save and Exit");
               System.out.print("Choose an option: ");
               String choice = scanner.nextLine();
               // You will handle the choice entered here using switch(case)
           }
       }
       // Add the methods to handle each choice here
   ```

Add the methods to handle user input

1. Add the following method to `EcoPointsRecyclingTracker.java` after the `main` method. It registers a new household in the Eco-Points Recycling Tracker. This method collects input from the user, validates the ID, creates a Household object, and adds it to the household's map.

   ```
       // Task 3
       private static void registerHousehold() {
           // Prompt the user to enter a unique household ID
           System.out.print("Enter household ID: ");
           String id = scanner.nextLine().trim();  // Read and trim input
           // Check if a household with this ID already exists in the map
           if (households.containsKey(id)) {
               System.out.println("Error: Household ID already exists.");
               return;  // Stop and return early if duplicate found
           }
           // Prompt the user to enter the household's name
           System.out.print("Enter household name: ");
           String name = scanner.nextLine().trim();
           // Prompt the user to enter the household's address
           System.out.print("Enter household address: ");
           String address = scanner.nextLine().trim();
           // Create a new Household object using the provided details
           Household household = new Household(id, name, address);
   ```

```
        // Add the new household to the households map (using ID as the key)
        households.put(id, household);
        // Confirm to the user that the household was registered successfully
        System.out.println("Household registered successfully on " + household.getJoinDate());
    }
```

2. Add the following method to `EcoPointsRecyclingTracker.java` after the `registerHousehold` method to log a new recycling event for an existing household.

   This method asks for the household ID and validates the household ID. Then the method collects recycling details, validates the recycling details, creates a RecyclingEvent, and updates the household record.

```
    // Task 4
    private static void logRecyclingEvent() {
        // Ask the user for the household ID
        System.out.print("Enter household ID: ");
        String id = scanner.nextLine().trim();
        // Look up the household in the map by ID
        Household household = households.get(id);
        // If household not found, show error and exit
        if (household == null) {
            // Task 8
            System.out.println("Error: Household ID not found.");
            return;
        }
        // Ask the user for the material type they recycled
        System.out.print("Enter material type (plastic/glass/metal/paper): ");
        String material = scanner.nextLine().trim();
        double weight = 0.0;
        // Loop until a valid weight is entered
        while (true) {
            try {
                System.out.print("Enter weight in kilograms: ");
                weight = Double.parseDouble(scanner.nextLine());  // Convert input to double
                // Check that weight is a positive number
                if (weight <= 0) throw new IllegalArgumentException();
                break;  // Exit loop if input is valid
            }  catch (NumberFormatException e) {
                System.out.println("Invalid weight. Must be a positive number.");
            }  catch (IllegalArgumentException e) {
                System.out.println("Invalid weight. Must be a positive number.");
            }
        }
        // Create a new RecyclingEvent using the material and weight
        RecyclingEvent event = new RecyclingEvent(material, weight);
        // Add the new event to the household and update points
        household.addEvent(event);
        // Show success message with points earned
        System.out.println("Recycling event logged! Points earned: " + event.getEcoPoints());
    }
```

3. Add the following method to `EcoPointsRecyclingTracker.java` after the `logRecyclingEvent` method. This method displays a list of all registered households. If no households exist, the method displays a message and exits.

```
    // Task 6
    private static void displayHouseholds() {
        // Check if the households map is empty
        if (households.isEmpty()) {
            System.out.println("No households registered.");
            return; // Exit early if there's nothing to show
        }
        // If there are households, print a header first
        System.out.println("\nRegistered Households:");
        // Loop through each household in the map and print its details
        for (Household h : households.values()) {
            System.out.println("ID: " + h.getId() +
                               ", Name: " + h.getName() +
                               ", Address: " + h.getAddress() +
                               ", Joined: " + h.getJoinDate());
        }
    }
```

4. Add the following method to `EcoPointsRecyclingTracker.java` after the `displayHouseholds` method. This method displays all recycling events for a specific household. This method displays the total weight and total points earned as well.

```
    // Task 6
    private static void displayHouseholdEvents() {
        // Prompt the user to enter the household ID
        System.out.print("Enter household ID: ");
        String id = scanner.nextLine().trim();
        // Look up the household in the households map using the ID
        Household household = households.get(id);
        // If household is not found, show an error and exit
        if (household == null) {
            System.out.println("Household not found.");
            return;
        }
        // Print a header with the household's name
        System.out.println("\nRecycling Events for " + household.getName() + ":");
        // Check if the household has any recycling events
        if (household.getEvents().isEmpty()) {
            System.out.println("No events logged.");
        } else {
            // Loop through all recycling events and print each one
            for (RecyclingEvent e : household.getEvents()) {
                //Print the stringified version of the event
                System.out.println(e);
            }
            // After listing events, show the total weight recycled by this household
            System.out.println("Total Weight: " + household.getTotalWeight() + " kg");
            // Show the total eco points earned by this household
            System.out.println("Total Points: " + household.getTotalPoints() + " pts");
        }
    }
```

5. Add the following method to `EcoPointsRecyclingTracker.java` after the `displayHouseholdEvents` method. This method generates simple reports for the Eco-Points Recycling Tracker. This method displays the following information:

    - The household with the highest total eco points.
    - The total community recycling weight.

```
    // Task 7
    private static void generateReports() {
        // Check if there are any households registered
        if (households.isEmpty()) {
            System.out.println("No households registered.");
            return; // Exit if there's nothing to report on
        }
        // ------------------------------
        // Find the household with the highest points
        // ------------------------------
        Household top = null; // Start with no top household
        for (Household h : households.values()) {
            // If 'top' is still null, or this household has more points, update 'top'
            if (top == null || h.getTotalPoints() > top.getTotalPoints()) {
                top = h;
            }
        }
        // Print details of the top household
        System.out.println("\nHousehold with Highest Points:");
        System.out.println("ID: " + top.getId() +
                           ", Name: " + top.getName() +
                           ", Points: " + top.getTotalPoints());
        // ------------------------------
        // Calculate total community recycling weight
        // ------------------------------
        double totalWeight = 0.0;
        // Loop through all households to sum up their total weights
        for (Household h : households.values()) {
            totalWeight += h.getTotalWeight();
        }
        // Print total community weight
        System.out.println("Total Community Recycling Weight: " + totalWeight + " kg");
    }
```

This code performs the following tasks:

- Loops over households.values() to process each Household.
- Locates a "top" item using compare in a loop. If there is a top item already assigned, updates the "top" item if the current iteration's points are greater.
- Performs and aggregate calculation that sums values from multiple objects (getTotalWeight()).
- Displays neatly formatted results make reports readable and useful.

6. Add the following method to `EcoPointsRecyclingTracker.java` after the `generateReports` method. This method saves all household data to a file using serialization. This method ensures that households and their recycling events persist even after the program closes.

```
    // Task 5
    private static void saveHouseholdsToFile() {
        try {
            // Create a FileOutputStream to write to the file named "households.ser"
            ObjectOutputStream out = new ObjectOutputStream(
                new FileOutputStream("households.ser")
            );
            // Write the entire households map to the file
            out.writeObject(households);
            // If successful, no message is printed here — could add confirmation if you like
        } catch (IOException e) {
            // Task 8
            // If something goes wrong while saving, print an error message
            System.out.println("Error saving data: " + e.getMessage());
        }
    }
```

Here's how the code works:

- Serialization converts your HashMap of Household objects into a byte stream that can be saved on disk.
- ObjectOutputStream is a special stream to write whole objects, not just text.
- `try-catch` makes sure that file problems don't crash the app.

7. Add the following method to `EcoPointsRecyclingTracker.java` after the `saveHouseholdsToFile` method. This method loads all household data from a file using deserialization. If the file exists and contains data, there is Household data that was saved previously. This method restores the households map. If the file doesn't exist, the program starts with an empty map.

```
@SuppressWarnings("unchecked") // Suppresses unchecked cast warning when reading the object
private static void loadHouseholdsFromFile() {
    // Use a try-with-resources block to automatically close the input stream
    try (
        // Open an ObjectInputStream to read from the file "households.ser"
        ObjectInputStream in = new ObjectInputStream(new FileInputStream("households.ser"))
    ) {
        // Read the object from the file and cast it back to the correct type
        households = (Map<String, Household>) in.readObject();
        // Confirmation message to let the user know data was loaded
        System.out.println("Household data loaded.");
    } catch (FileNotFoundException e) {
        // Task 8
        // If the file doesn't exist yet, that's okay – start with empty data
        System.out.println("No saved data found. Starting fresh.");
    } catch (IOException | ClassNotFoundException e) {
        // Handle other errors, like if the file is corrupted or unreadable
        System.out.println("Error loading data: " + e.getMessage());
    }
}
```

Add code to main method

1. Inside the main method, as the first action, the method loads any existing saved records before getting into the `while` loop to validate user input.

```
loadHouseholdsFromFile();
```

2. Next, inside the method, in the space provided, you will need to add the switch-case statement to handle the user choice input and call the methods accordingly.

```
        switch (choice) {
            case "1":
                registerHousehold();
                break;
            case "2":
                logRecyclingEvent();
                break;
            case "3":
                displayHouseholds();
                break;
            case "4":
                displayHouseholdEvents();
                break;
            case "5":
                generateReports();
                break;
            case "6":
                saveHouseholdsToFile();
                running = false;
                System.out.println("Data saved. Goodbye!");
                break;
            default:
                System.out.println("Invalid choice. Please select 1-6.");
        }
```

Here's how the code works:

- `switch` looks at the value of the choice variable (a String that came from user input).
- If the user typed "1", run the registerHousehold() method.
- If the user typed "2", run the logRecyclingEvent() method.
- If the user typed "3", run the displayHouseholds() method.
- If the user typed "4", run the displayHouseholdEvents() method.
- If the user typed "5", run the generateReports() method.
- If the user typed "6", run the saveHouseholdsToFile() method.
- If the user typed any other key, they get a helpful error message.

▶ Click here to look at the entire solution

Compile and run

1. Now that you have assimilated all the code together, you can compile and run the files. Ensure that you are still in the project directory.

   ```
   cd /home/project/EcoPointsRecyclingTracker
   ```

2. Compile all the classes by running the following command.

   ```
   javac *.java
   ```

3. Run the application, by executing the following.

   ```
   java EcoPointsRecyclingTracker
   ```

4. You will see an output depending on your choice. A sample output has been provided. Ensure that you try out all the different user choices.

```
=== Eco-Points Recycling Tracker ===
1. Register Household
2. Log Recycling Event
3. Display Households
4. Display Household Recycling Events
5. Generate Reports
6. Save and Exit
Choose an option: 1
Enter household ID: 1
Enter household name: Srirams
Enter household address: 31 Vears Road, Ashburton
Household registered successfully on 2025-07-06
=== Eco-Points Recycling Tracker ===
1. Register Household
2. Log Recycling Event
3. Display Households
4. Display Household Recycling Events
5. Generate Reports
6. Save and Exit
Choose an option: 1
Enter household ID: 2
Enter household name: Cootes
Enter household address: 14 Liston street Glen Iris
Household registered successfully on 2025-07-06
=== Eco-Points Recycling Tracker ===
1. Register Household
2. Log Recycling Event
3. Display Households
4. Display Household Recycling Events
5. Generate Reports
6. Save and Exit
Choose an option: 2
Enter household ID: 1
Enter material type (plastic/glass/metal/paper): plastic
Enter weight in kilograms: 2
Recycling event logged! Points earned: 20.0
=== Eco-Points Recycling Tracker ===
1. Register Household
2. Log Recycling Event
3. Display Households
4. Display Household Recycling Events
5. Generate Reports
6. Save and Exit
Choose an option: 2
Enter household ID: 1
Enter material type (plastic/glass/metal/paper): glass
Enter weight in kilograms: 3
Recycling event logged! Points earned: 30.0
=== Eco-Points Recycling Tracker ===
1. Register Household
2. Log Recycling Event
3. Display Households
4. Display Household Recycling Events
5. Generate Reports
6. Save and Exit
Choose an option: 3
Registered Households:
ID: 1, Name: Srirams, Address: 31 Vears Road, Ashburton, Joined: 2025-07-06
ID: 2, Name: Cootes, Address: 14 Liston street Glen Iris, Joined: 2025-07-06
=== Eco-Points Recycling Tracker ===
1. Register Household
2. Log Recycling Event
```

```
3. Display Households
4. Display Household Recycling Events
5. Generate Reports
6. Save and Exit
Choose an option: 4
Enter household ID: 1
Recycling Events for Srirams:
2025-07-06 | plastic | 2.0 kg | 20.0 pts
2025-07-06 | glass | 3.0 kg | 30.0 pts
Total Weight: 5.0 kg
Total Points: 50.0 pts
=== Eco-Points Recycling Tracker ===
1. Register Household
2. Log Recycling Event
3. Display Households
4. Display Household Recycling Events
5. Generate Reports
6. Save and Exit
Choose an option:
```

Checklist for tasks

At this point all of these tasks are accomplished.

- Task 1: Household and RecyclingEvent Classes created
- Task 2: ArrayList to store Recycling Events and HashMap to store Households are created
- Task 3: Households can be registered.
- Task 4: Recycling Events can be logged for households.
- Task 5: Data can be stored. Data can be loaded from stored file.
- Task 6: Records displayed as per all the registered households, all recycling events for a household, total weight recycled by a household and total eco points earned by a household.
- Task 7: Reports can be generated.
- Task 8: Error Handling is implemented to catch invalid inputs (like negative weights), handle duplicate household IDs, manage file read/write errors.

Congratulations!

To summarize, by completing the Eco-Points Recycling Tracker, you've learned how to:

- Write clear, well-structured Java code using variables, operators, and data types.
- Handle user input using Scanner and process String data safely.
- Use if statements and switch blocks to make your program respond to user choices.
- Use `while` and `for` loops to repeat tasks until they're done correctly or to process lists of data.
- Design your own classes following the object-oriented programming (OOP) concepts to represent real-life data.
- Use ArrayList to store multiple recycling events for each household.
- Use HashMap to manage and quickly access many households using unique IDs.
- Use the modern Java Date and Time API (LocalDate) to record and display dates automatically.
- Use Java serialization to write and read complex data structures (like your whole HashMap of households) to and from files.
- Understand how file input/output helps real programs remember data between sessions.
- Use try-catch blocks to handle invalid input, duplicate IDs, or file errors without crashing your program.

Author(s)

Lavanya