# INTERNSHIP (JAVA DEVELOPMENT)

| | | |
|---|---|---|
| NAME | : | DEEPIKA G |
| REGISTER NUMBER | : | 732424104016 |
| DEGREE | : | BACHELOR OF EDUCATION |
| DEPARTMENT | : | COMPUTER SCIENCE AND ENGINEERING |
| DATE | : | 19-12-2025 |

BASIC LEVEL PROJECT :

Project Title: Daily Habit Tracker (Console Application)

CODE:

```
package healthhabits;
import java.util.Scanner;
 public class HabitTrackers {
static Scanner sc = new Scanner(System.in);
 static String[] habits = new String[10];
 static boolean[] completed = new boolean[10];
static int count = 0;
static int weeklyScore = 0;
public static void main(String[] args) {
 int choice;
 do {
 System.out.println("\n=== Daily Habit Tracker ===");
System.out.println("1. Add Habit");
System.out.println("2. Mark Habit Completed");
System.out.println("3. View Habits");
System.out.println("4. Weekly Score");
 System.out.println("5. Exit");
 System.out.print("Enter choice: ");
choice = sc.nextInt();
```

```
sc.nextLine();
 if (choice == 1) addHabit();
else if (choice == 2) markHabit();
else if (choice == 3) viewHabits();
else if (choice == 4) showScore();
 } while (choice != 5);
System.out.println("Goodbye! Keep building good habits ");
}
static void addHabit() {
     System.out.print("Enter habit name: ");
habits[count] = sc.nextLine();
completed[count] = false;
count++;
System.out.println("Habit added!");
}
static void markHabit() {
   viewHabits();
  System.out.print("Enter habit number to mark completed: ");
  int index = sc.nextInt();
 if (index > 0 && index <= count) {
     completed[index - 1] = true;
     weeklyScore++;
System.out.println("Habit marked completed!");
 } else {
```

```
 System.out.println("Invalid number.");
   }
 }
 static void viewHabits() {
if (count == 0) {
   System.out.println("No habits yet.");
} else {
   for (int i = 0; i < count; i++) {
System.out.println((i + 1) + ". " + habits[i] + " -  " +
(completed[i] ?       "Completed " : "Pending "));
     }
  }
}
static void showScore() {
 System.out.println("Weekly Habit Score: " + weeklyScore);
    }
  }
```

OUTPUT:

```
 === Daily Habit Tracker ===
1. Add Habit
 2. Mark Habit Completed
3. View Habits
 4. Weekly Score
```

```
5. Exit
Enter choice: 1
 Enter habit name: Singing
 Habit added!
 === Daily Habit Tracker ===
1. Add Habit
2. Mark Habit Completed
3. View Habits
4. Weekly Score
5. Exit
Enter choice: 1
Enter habit name: Playing Habit added!
 === Daily Habit Tracker ===
1. Add Habit
2. Mark Habit Completed
3. View Habits
4. Weekly Score
5. Exit
Enter choice: 1
 Enter habit name: Eating
Habit added!
=== Daily Habit Tracker ===
1. Add Habit
 2. Mark Habit Completed
 3. View Habits
 4. Weekly Score
5. Exit
 Enter choice: 2
 1. Singing - Pending
2. Playing - Pending
3. Eating – Pending
 Enter habit number to mark completed: 1
 Habit marked completed!
=== Daily Habit Tracker ===
    1.   Add Habit
    2. Mark Habit Completed
    3. View Habits
    4. Weekly Score
     5. Exit
    Enter choice: 3
    1. Singing - Completed
    2. Playing - Completed
    3. Eating – Completed
     === Daily Habit Tracker ===
    1. Add Habit
    2. Mark Habit Completed
    3. View Habits
     4. Weekly Score
     5. Exit
    Enter choice: 4
    Weekly Habit Score: 3
    === Daily Habit Tracker ===
    1. Add Habit
    2. Mark Habit Completed
     3. View Habits
    4. Weekly Score
     5. Exit
    Enter choice: 5
    Goodbye! Keep building good habits
```

MEDIUM LEVEL PROJECT

Project Title: Expense Splitter (Mini Splitwise Console App)

CODE:

```
   package expense;
   import java.util.*;
   public class amount {
      static class User {
          int id;
         String name;
         User(int id, String name) { this.id = id; this.name = name; }
         public String toString() { return id + " - " + name; }
 }
 static class Expense {
   int payerId;
 double amount;
 String description;
List participants;
Expense(int payerId, double amount, String description, List
participants) {
 this.payerId = payerId;
 this.amount = amount;
 this.description = description;
 this.participants = participants;
 }
 public String toString() {
    return "Expense: " + description + " | Amount: " + amount + " |
Payer: " + payerId + " | Participants: " + participants;
   }
}
static class ExpenseManager {
    int nextUserId = 1;
    Map<Integer,User> users = new HashMap<>();
   List< Expenses>expenses = new ArrayList<>();
   Map<Integer,Double balances = new HashMap<>();
  User addUser(String name) {
    User u = new User(nextUserId, name);
    users.put(nextUserId, u);
    balances.put(nextUserId, 0.0);
   nextUserId++;
  System.out.println("Added: " + u);
 return u;
 }
void addExpense(int payerId, double amount, String desc, List
participants) {
     if (participants == null || participants.isEmpty()) {
       System.out.println("Error: participants cannot be empty.");
 return;
}
```

```
 if (!users.containsKey(payerId)) {
System.out.println("Error: payer id " + payerId + " does not
exist.");  return;
}
 for (int pid : participants) {
   if (!users.containsKey(pid)) {
       System.out.println("Error: participant id " + pid + " does not
exist.");
        return;
   }
}
 Expense e = new Expense(payerId, amount, desc,
participants); expenses.add(e);
 double share = amount / participants.size();
for (int pid : participants) {
      balances.put(payerId, balances.getOrDefault(payerId, 0.0)
+ share);
      balances.put(pid, balances.getOrDefault(pid, 0.0) - share);
 }
System.out.println("Expense added: " + e);
}
void showUsers() {
     if (users.isEmpty()) { System.out.println("No users.");
return; }
    System.out.println("Users:");
    for (User u : users.values()) System.out.println(" -  " + u);
}
void showExpenses() {
  if (expenses.isEmpty()) { System.out.println("No expenses.");
return; } System.out.println("Expenses:");
 for (Expense e : expenses) System.out.println(" -  " + e);
 }
 void showSummary() {
if (balances.isEmpty()) { System.out.println("No balances
yet."); return; } System.out.println("\nSettlement Summary:");
 for (int id : balances.keySet()) {
    double bal = Math.round(balances.get(id) * 100.0) / 100.0;
     if (bal > 0) System.out.println(users.get(id).name + " should
RECEIVE ₹" + bal);
     else if (bal < 0) System.out.println(users.get(id).name + "
should PAY ₹" + (-bal));
    else System.out.println(users.get(id).name + " is SETTLED.");
      }
    }
}
public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
     ExpenseManager manager = new ExpenseManager();
     while (true) {
         System.out.println("\nMenu:");
```

```
        System.out.println("1. Add User");
       System.out.println("2. Add Expense");
      System.out.println("3. Show Users");
      System.out.println("4. Show Expenses");
     System.out.println("5. Show Summary");
     System.out.println("6. Exit");
     System.out.print("Choose: ");
     if (!sc.hasNextInt()) { sc.nextLine();
System.out.println("Invalid choice."); continue; }
      int choice = sc.nextInt();
     sc.nextLine();
 switch (choice) {
    case 1 -> {
    System.out.print("Enter user name: ");
    String name = sc.nextLine().trim();
    if (name.isEmpty()) { System.out.println("Name cannot be
empty."); break; }
    manager.addUser(name);
}
  case 2 -> {
  System.out.print("Enter payer id: ");
  if (!sc.hasNextInt()) { sc.nextLine();
System.out.println("Invalid payer id."); break; }
    int payer = sc.nextInt();
   System.out.print("Enter amount: ");
```

```
    if (!sc.hasNextDouble()) { sc.nextLine();
System.out.println("Invalid amount."); break; }
         double amt = sc.nextDouble();
         sc.nextLine();
        System.out.print("Enter description: ");
       String desc = sc.nextLine().trim();
       System.out.print("Enter participant ids (comma
separated): ");
      String[] parts = sc.nextLine().split(",");
       List pids = new ArrayList<>();
       for (String s : parts) {
            s = s.trim();
            if (s.isEmpty()) continue;
           try { pids.add(Integer.parseInt(s)); }
          catch (NumberFormatException ex)
{ System.out.println("Invalid id: " + s); }
          }
         manager.addExpense(payer, amt, desc, pids);
}
 case 3 -> manager.showUsers();
case 4 -> manager.showExpenses();
case 5 -> manager.showSummary();
 case 6 -> {
         System.out.println("Exiting...");
         sc.close();
        return;
  }
       default -> System.out.println("Invalid choice!");
      }
    }
  }
}
```

Output:

```
Menu:
1. Add User
2. Add Expense
3. Show Users
4. Show Expenses
5. Show Summary
 6. Exit
Choose: 1
Enter user name: john
Added: 1 - joh
Menu:
1. Add User
 2. Add Expense
 3. Show Users
4. Show Expenses
 5. Show Summary
 6. Exit
Choose: 1
 Enter user name: Daniel
 Added: 2 – daniel
 Menu:
 1. Add User
 2. Add Expense
 3. Show Users
4. Show Expenses
 5. Show Summary
6. Exit
Choose: 2
Enter payer id: 1
 Enter amount: 2000
Enter description: lunch
Enter participant ids (comma separated): 1,2
Expense added: Expense: lunch | Amount: 2000.0 | Payer: 1 |
   Participants: [1, 2]
Menu:
 1. Add User
 2. Add Expense
3. Show Users
 4. Show Expenses
5. Show Summary
```

```
6. Exit
 Choose: 3
Users:
 - 1 – john
 - 2 - daniel
Menu:
1. Add User
2. Add Expense
 3. Show Users
4. Show Expenses
5. Show Summary
6. Exit
Choose: 4
 Expenses:
 - Expense: lunch | Amount: 2000.0 | Payer: 1 | Participants: [1,
2] Menu:
 1. Add User
 2. Add Expense
 3. Show Users
4. Show Expenses
5. Show Summary
 6. Exit
Choose: 5
 Settlement Summary:
```

```
john should RECEIVE ₹1000.0
daniel should PAY ₹1000.0
Menu:
 1. Add User
 2. Add Expense
 3. Show Users
4. Show Expenses
5. Show Summary
6. Exit
Choose: 6
 Exiting...
```

ADVANCED LEVEL PROJECT

Project Title: Smart Parking Lot System

CODE:

```
    package Vehicle;
    import java.io.*;
   import java.time.*;
   import java.util.*;
 public class SmartParkingSystem {
     static class NoSlotAvailableException extends Exception {
```

```
  public NoSlotAvailableException(String msg) { super(msg); }
  }
 static class VehicleNotFoundException extends Exception {
   public VehicleNotFoundException(String msg) { super(msg); }
  }
static class Vehicle {
     String plate;
    String type;
    LocalDateTime entryTime;
   Vehicle(String plate, String type) {
      this.plate = plate;
      this.type = type;
      this.entryTime = LocalDateTime.now();
   }
 }
static class Ticket {
    Vehicle vehicle;
    int slotId;
   LocalDateTime exitTime;
  double charges;
 Ticket(Vehicle v, int slotId) {
  this.vehicle = v;
this.slotId = slotId;
  }
  void checkout() {
     this.exitTime = LocalDateTime.now();

     Duration d = Duration.between(vehicle.entryTime, exitTime)
     long hours = Math.max(1, d.toMinutes()/60); // minimum
1 hr
    charges = calculateCharges(vehicle.type, hours);
 }
private double calculateCharges(String type, long hours) {
 switch(type) {
    case "CAR": return hours * 40;
    case "BIKE": return hours * 20;
    case "TRUCK": return hours * 75;
   default: return hours * 30;
   }
}
public String toString() {
    return "Ticket: Plate=" + vehicle.plate +
              ", Type=" + vehicle.type +
              ", Slot=" + slotId +
              ", Entry=" + vehicle.entryTime +
             ", Exit=" + exitTime +
             ", Charges=₹" + charges;
    }
 }
```

```
static class ParkingLot {
      int totalSlots;
     Map activeTickets = new HashMap<>();
    Set occupiedSlots = new HashSet<>();
    ParkingLot(int slots) { this.totalSlots = slots; }
    void addVehicle(String plate, String type) throws
     NoSlotAvailableException
{
        int slotId = findFreeSlot();
        if(slotId == -1) throw new NoSlotAvailableException("No
free slots available!");
            Vehicle v = new Vehicle(plate, type);
            Ticket t = new Ticket(v, slotId);
            activeTickets.put(plate, t);
           occupiedSlots.add(slotId);
        System.out.println("Vehicle parked: " + plate + " at Slot " +
slotId);
        saveToFile("tickets.txt", "PARKED: " + t);
}
 void removeVehicle(String plate) throws
VehicleNotFoundException {
  if(!activeTickets.containsKey(plate)) throw new
VehicleNotFoundException("Vehicle not found!");
      Ticket t = activeTickets.get(plate);
      t.checkout
     occupiedSlots.remove(t.slotId);
    activeTickets.remove(plate);
  System.out.println("Vehicle removed: " + plate);
  System.out.println(t);
  saveToFile("tickets.txt", "REMOVED: " + t);
 }
  int findFreeSlot() {
  for(int i=1;i<=totalSlots;i++) {
  if(!occupiedSlots.contains(i)) return i;
  }
  return -1;
}
void showStatus() {
    System.out.println("Total Slots: " + totalSlots);
    System.out.println("Occupied: " + occupiedSlots.size());
    System.out.println("Free: " + (totalSlots -
occupiedSlots.size()));
}
void saveToFile(String filename, String data) {
  try(FileWriter fw = new FileWriter(filename, true)) {
        fw.write(data + "\n");
} catch(IOException e) {
       System.out.println("File error: " + e.getMessage());
     }
```

```
   }
 }
  public static void main(String[] args) {
    ParkingLot lot = new ParkingLot(5); // 5 slots
   Scanner sc = new Scanner(System.in);
   while(true) {
System.out.println("\n1.Add Vehicle 2.Remove Vehicle
3.Status 0.Exit");
      if (!sc.hasNextInt()) {
System.out.println("Enter a valid choice number.");
    sc.next();
    continue;
   }
  int choice = sc.nextInt();
  try {
  if(choice==1) {
       System.out.print("Enter plate: ");
      String plate = sc.next();
      System.out.print("Enter type (CAR/BIKE/TRUCK): ");
      String type = sc.next().toUpperCase();
      lot.addVehicle(plate, type);
} else if(choice==2) {
      System.out.print("Enter plate: ");
     String plate = sc.next();
     lot.removeVehicle(plate);
} else if(choice==3) {
      lot.showStatus();
} else if(choice==0) {
      break;
} else {
  System.out.println("Invalid choice.");
      }
  } catch(Exception e) {
     System.out.println("Error: " + e.getMessage());
     }
}
sc.close();
   }
}
```

Output:

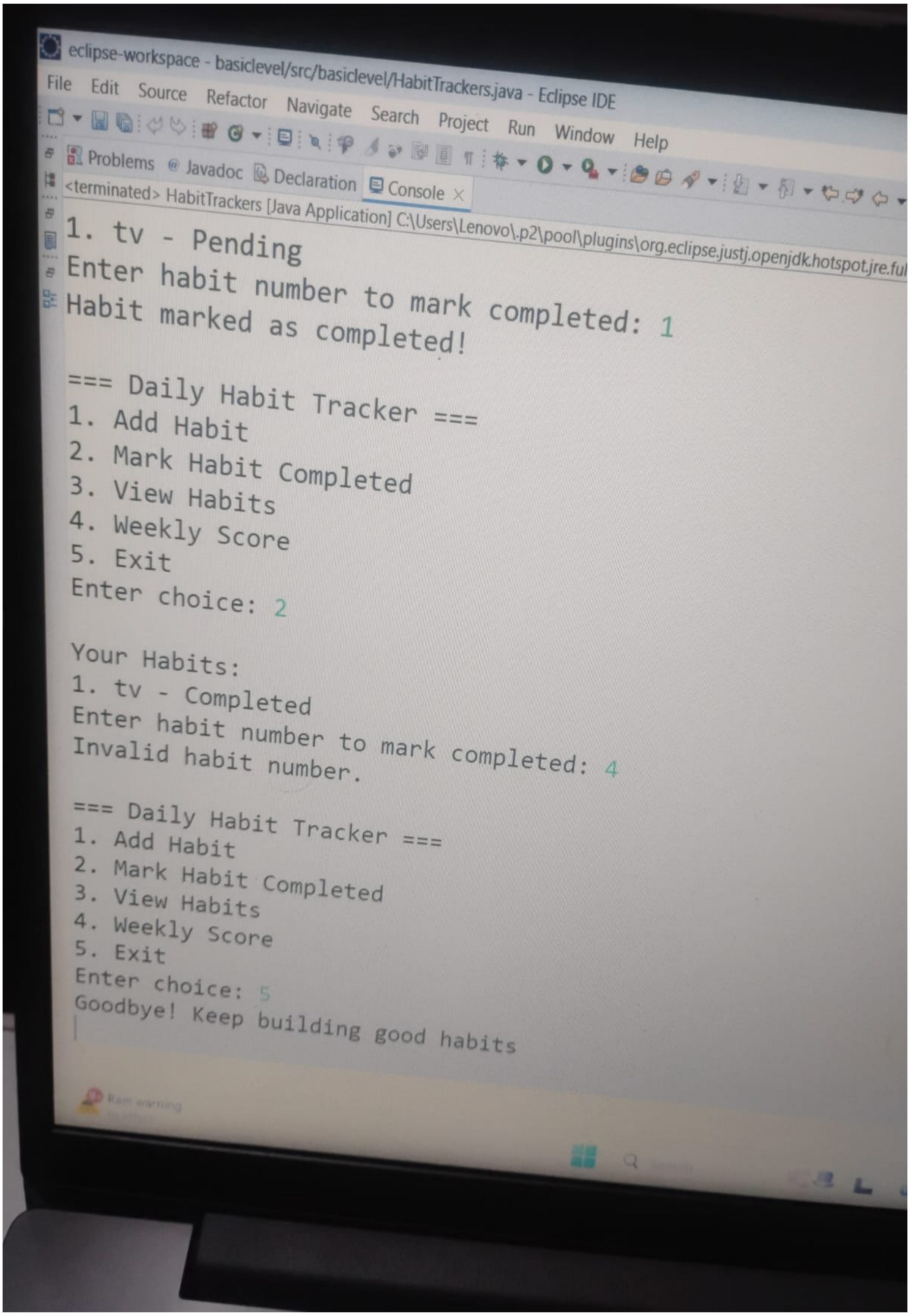
```
 1.Add Vehicle
2.Remove Vehicle
3.Status
0.Exit
1
 Enter plate: TN38BT5650
 Enter type (CAR/ BIKE/ TRUCK): Bike
 Vehicle parked: TN38BT5650 at Slot 1
```

```
--------------------------------------------------------------------
------------
 1.Add Vehicle
 2.Remove Vehicle
3.Status
0.Exit
1
 Enter plate: TN40DJ1201
Enter type (CAR/BIKE/TRUCK): Car
Vehicle parked: TN40DJ1201 at Slot 2

--------------------------------------------------------------------
-------------
1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit
1
Enter plate: TN39JD1201
Enter type (CAR/BIKE/TRUCK): Bike
Vehicle parked: TN39JD1201 at Slot 3
--------------------------------------------------------------------
--------------
1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit
3
Total Slots: 5
Occupied: 3
```

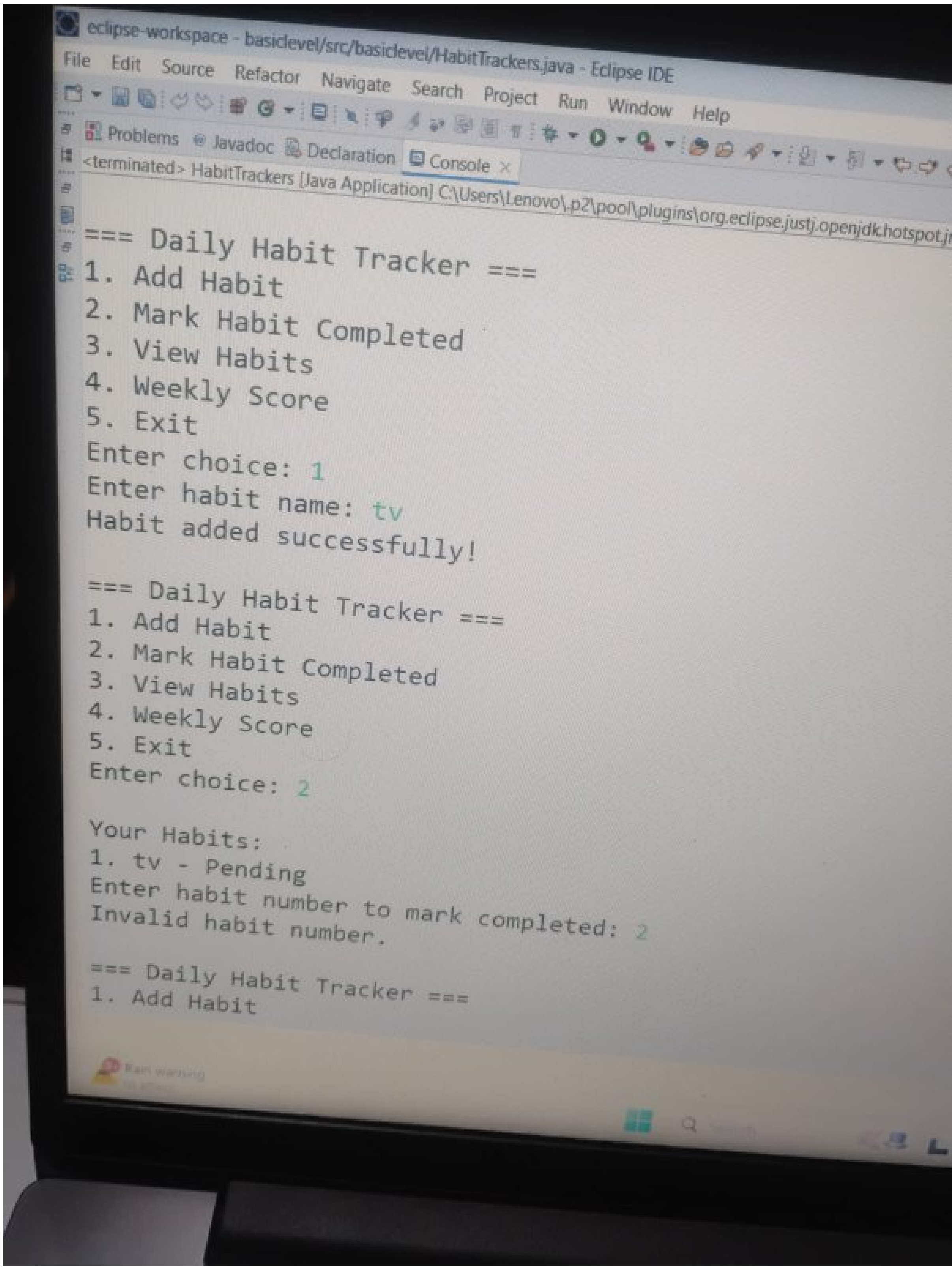
```
 Free: 2

-----------------------------------------------------------------
-----------
 1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit
1
 Enter plate: TN78YI7896
Enter type (CAR/BIKE/TRUCK): Car
Vehicle parked: TN78YI7896 at Slot 4

-----------------------------------------------------------------
----------
1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit
1
Enter plate: TN42KG2520
Enter type (CAR/BIKE/TRUCK): Bike
Vehicle parked: TN42KG2520 at Slot 5

-----------------------------------------------------------------
-----------
1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit
3
Total Slots: 5
 Occupied: 5
 Free: 0
```

```
------------------------------------------------------------------------
-----------

 1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit

2

Enter plate: TN42KG2520

Vehicle removed: TN42KG2520

Ticket: Plate=TN42KG2520, Type=BIKE, Slot=5,
Entry=2025-12-15T22:56:40.445411,
Exit=2025-12-15T22:57:05.154631400, Charges=₹20.0

------------------------------------------------------------------------
--------------

1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit

 3

Total Slots: 5

Occupied: 4

 Free: 1

1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit

 0
```

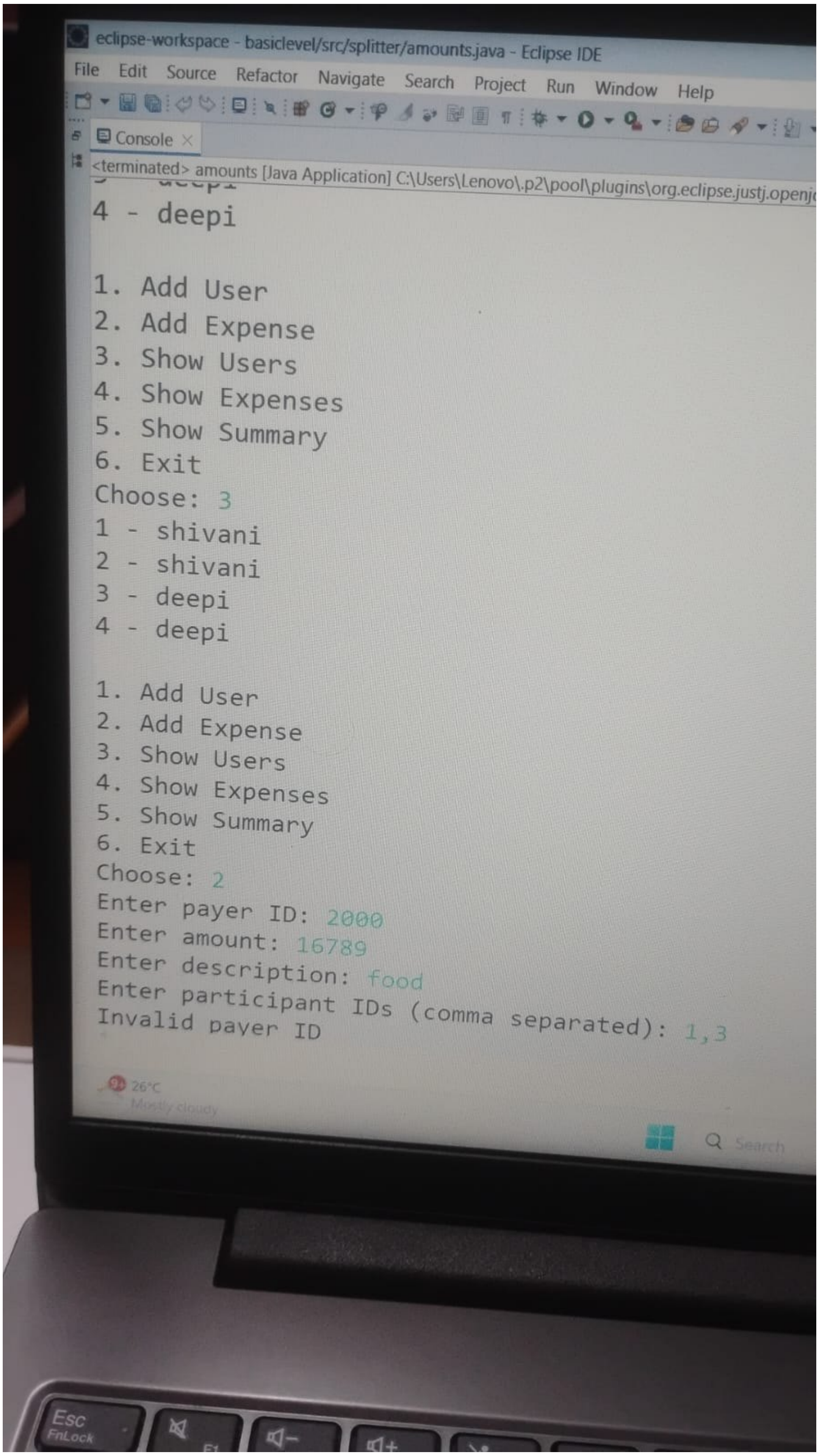
SCREEN SHORT :   BASIC LEVEL


eclipse-workspace - basiclevel/src/basiclevel/HabitTrackers.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console

```
<terminated> HabitTrackers [Java Application] C:\Users\Lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
1. tv - Pending
Enter habit number to mark completed: 1
Habit marked as completed!

=== Daily Habit Tracker ===
1. Add Habit
2. Mark Habit Completed
3. View Habits
4. Weekly Score
5. Exit
Enter choice: 2

Your Habits:
1. tv - Completed
Enter habit number to mark completed: 4
Invalid habit number.

=== Daily Habit Tracker ===
1. Add Habit
2. Mark Habit Completed
3. View Habits
4. Weekly Score
5. Exit
Enter choice: 5
Goodbye! Keep building good habits
```

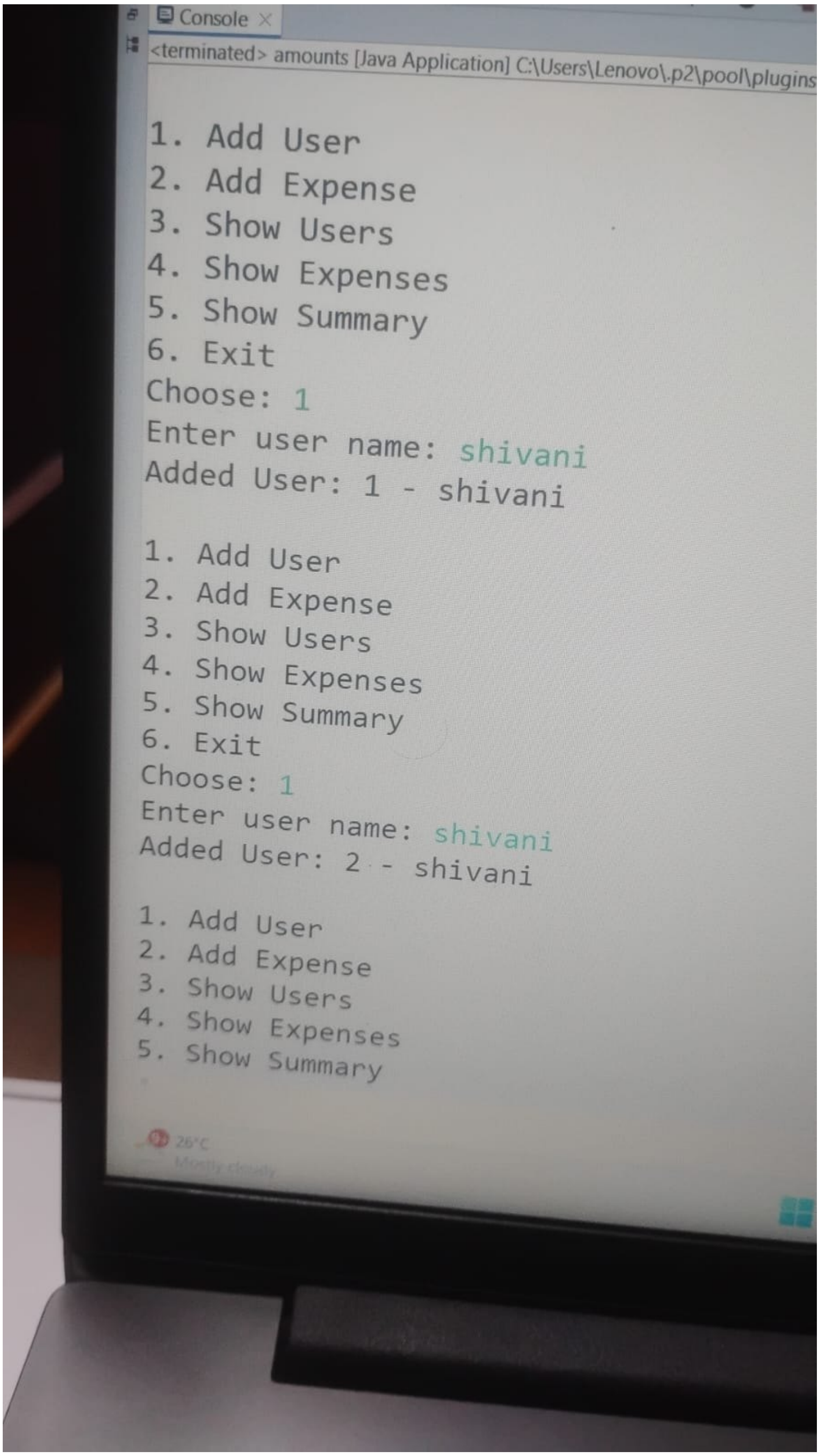

eclipse-workspace - basiclevel/src/basiclevel/HabitTrackers.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Problems Javadoc Declaration Console

```
<terminated> HabitTrackers [Java Application] C:\Users\Lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.j

=== Daily Habit Tracker ===
1. Add Habit
2. Mark Habit Completed
3. View Habits
4. Weekly Score
5. Exit
Enter choice: 1
Enter habit name: tv
Habit added successfully!

=== Daily Habit Tracker ===
1. Add Habit
2. Mark Habit Completed
3. View Habits
4. Weekly Score
5. Exit
Enter choice: 2

Your Habits:
1. tv - Pending
Enter habit number to mark completed: 2
Invalid habit number.

=== Daily Habit Tracker ===
1. Add Habit
```

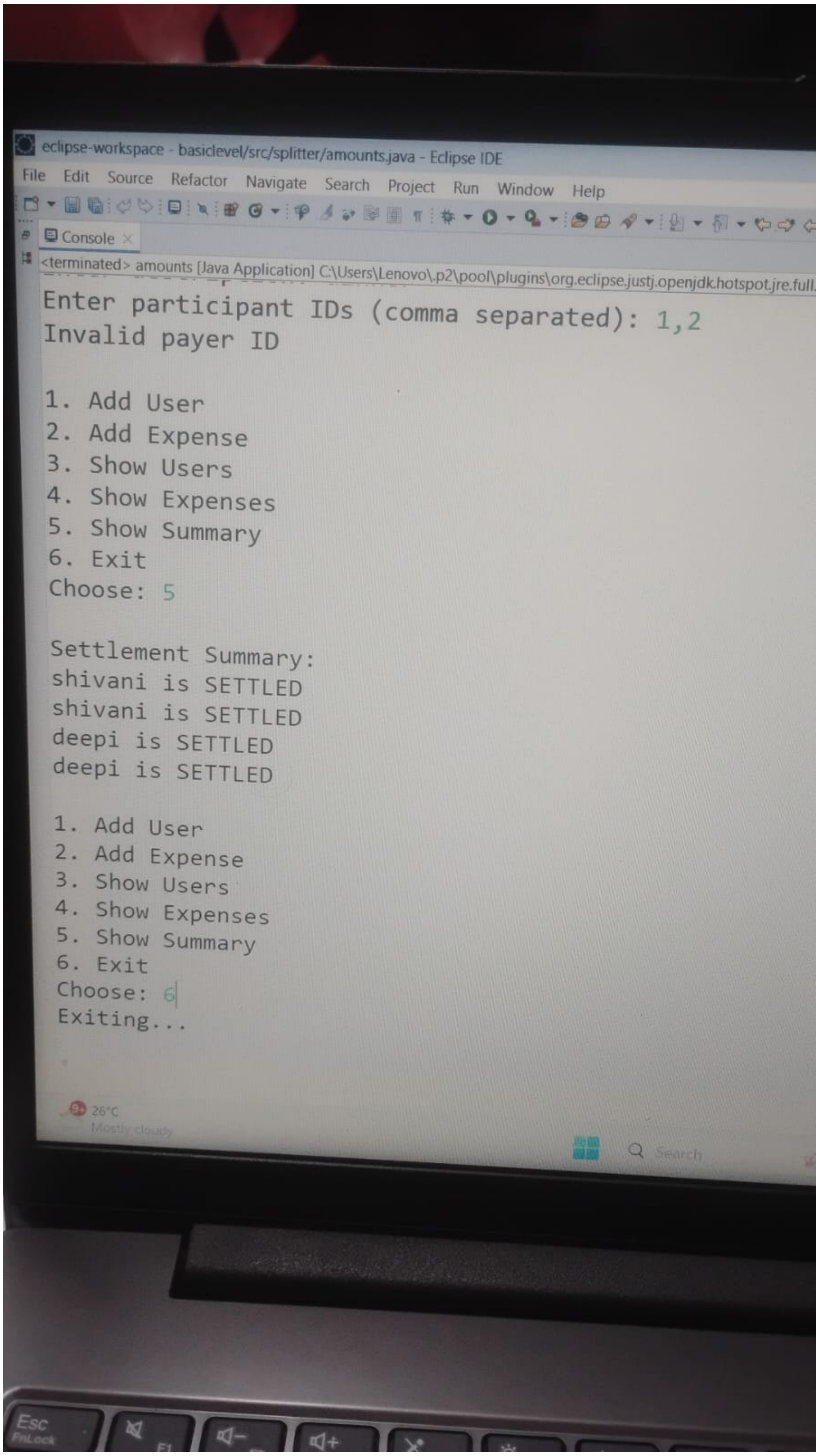


eclipse-workspace - basiclevel/src/splitter/amounts.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Console
<terminated> amounts [Java Application] C:\Users\Lenovo\.p2\pool\plugins\org.eclipse.justj.openj

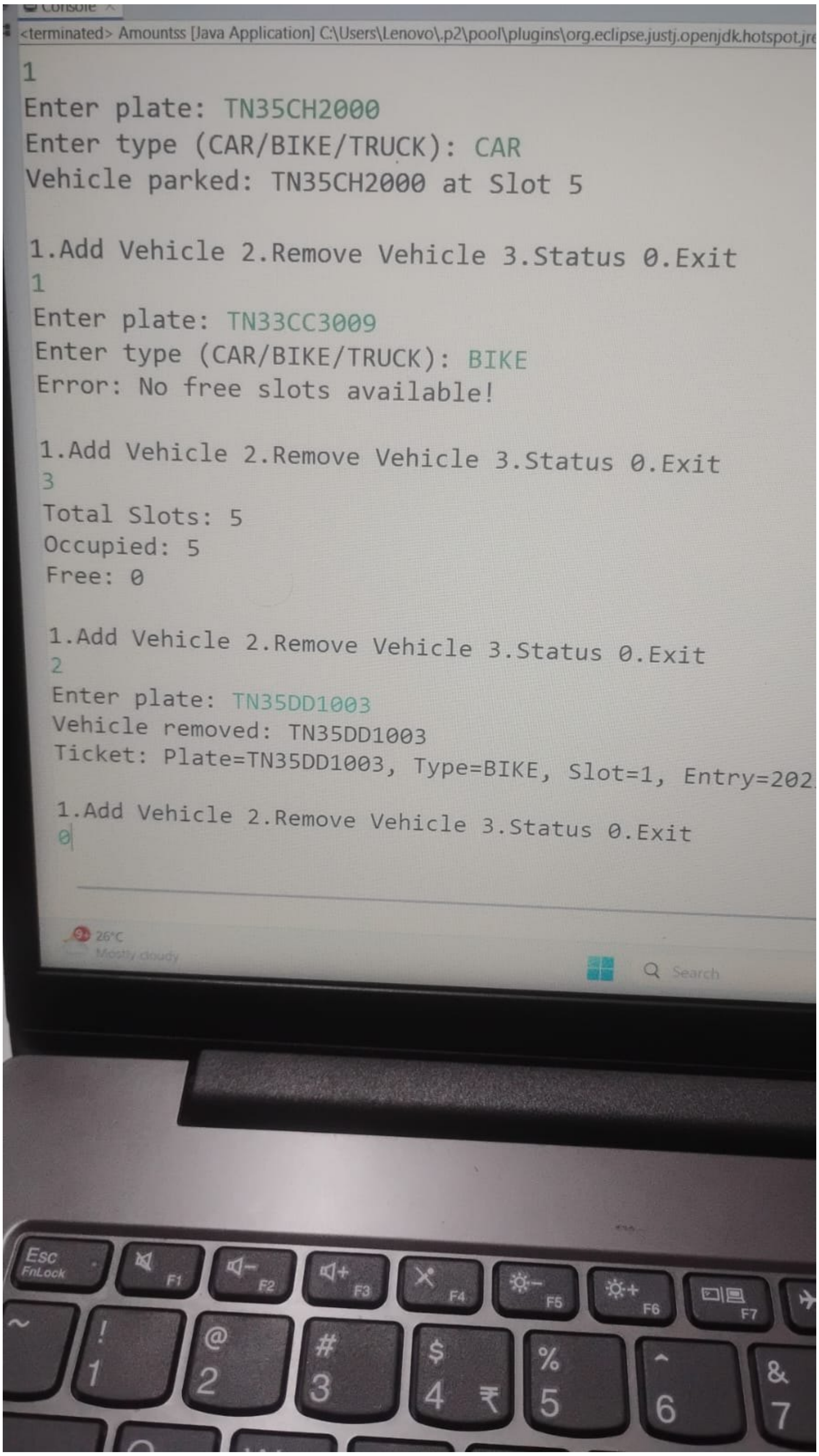
```
4 - deepi

1. Add User
2. Add Expense
3. Show Users
4. Show Expenses
5. Show Summary
6. Exit
Choose: 3
1 - shivani
2 - shivani
3 - deepi
4 - deepi

1. Add User
2. Add Expense
3. Show Users
4. Show Expenses
5. Show Summary
6. Exit
Choose: 2
Enter payer ID: 2000
Enter amount: 16789
Enter description: food
Enter participant IDs (comma separated): 1,3
Invalid payer ID
```



```
<terminated> amounts [Java Application] C:\Users\Lenovo\.p2\pool\plugins

1. Add User
2. Add Expense
3. Show Users
4. Show Expenses
5. Show Summary
6. Exit
Choose: 1
Enter user name: shivani
Added User: 1 - shivani

1. Add User
2. Add Expense
3. Show Users
4. Show Expenses
5. Show Summary
6. Exit
Choose: 1
Enter user name: shivani
Added User: 2 - shivani

1. Add User
2. Add Expense
3. Show Users
4. Show Expenses
5. Show Summary
```


eclipse-workspace - basiclevel/src/splitter/amounts.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Console
<terminated> amounts [Java Application] C:\Users\Lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.

```
Enter participant IDs (comma separated): 1,2
Invalid payer ID

1. Add User
2. Add Expense
3. Show Users
4. Show Expenses
5. Show Summary
6. Exit
Choose: 5

Settlement Summary:
shivani is SETTLED
shivani is SETTLED
deepi is SETTLED
deepi is SETTLED

1. Add User
2. Add Expense
3. Show Users
4. Show Expenses
5. Show Summary
6. Exit
Choose: 6
Exiting...
```



```
<terminated> Amountss [Java Application] C:\Users\Lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jr
1
Enter plate: TN35CH2000
Enter type (CAR/BIKE/TRUCK): CAR
Vehicle parked: TN35CH2000 at Slot 5

1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit
1
Enter plate: TN33CC3009
Enter type (CAR/BIKE/TRUCK): BIKE
Error: No free slots available!

1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit
3
Total Slots: 5
Occupied: 5
Free: 0

1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit
2
Enter plate: TN35DD1003
Vehicle removed: TN35DD1003
Ticket: Plate=TN35DD1003, Type=BIKE, Slot=1, Entry=202

1.Add Vehicle 2.Remove Vehicle 3.Status 0.Exit
0
```

THANKYOU