

# PROJECT TITLE

Mini Project Report

Object Oriented Programming (01CE0104)

Semester 2

Student Name – Khushaliba Rathod

Enrollment Number- 92400103533

Marwadi University

Department of Computer Engineering

Rajkot, Gujarat

April, 2025



**Marwadi**  
University  
Marwadi Chandarana Group



## Project Description:

Car Parking System is a console-based Java application developed using Object-Oriented Programming principles. The project aims to simulate a simple parking management system where users can enter, view, and reset the count of vehicles parked.

The system provides a menu-driven interface that allows users to manage the parking of different types of vehicles such as cars, bikes, and rickshaws. Users can input the type of vehicle entering the parking, view the current number of each type of vehicle, or reset all values when needed.

The core functionality is encapsulated within two classes: `Carparking` and `parking`. The `Carparking` class handles user input and the interface logic, while the `parking` class manages the state of the parking lot using private variables and accessor methods. This separation of concerns showcases good object-oriented design.

This project is ideal for learning basic OOP principles such as encapsulation, class interaction, and input handling. It can be further enhanced with features like time-based billing, vehicle number registration, and GUI integration for a complete real-world simulation.

The application provides features like customizable budget planning, automated expense tracking, and visual reports such as pie charts and bar graphs to illustrate spending habits and trends. Users can set monthly spending limits, receive notifications when they approach or exceed their budgets, and track their progress toward financial goals such as saving for a vacation, paying off debt, or building an emergency fund.

Data security and privacy are prioritized, ensuring that all financial records are safely stored and accessible only to the user. The tracker can also generate summaries and downloadable reports for better financial planning and analysis.

Car Parking System is a console-based Java application developed using Object-Oriented Programming principles. The project aims to simulate a simple parking management system where users can enter, view, and reset the count of vehicles parked.

The system provides a menu-driven interface that allows users to manage the parking of different types of vehicles such as cars, bikes, and rickshaws. Users can input the type of vehicle entering the parking, view the current number of each type of vehicle, or reset all values when needed.

The core functionality is encapsulated within two classes: `Carparking` and `parking`. The `Carparking` class handles user input and the interface logic, while the `parking` class manages the state of the parking lot using private variables and accessor methods. This separation of concerns showcases good object-oriented design.

This project is ideal for learning basic OOP principles such as encapsulation, class interaction, and input handling. It can be further enhanced with features like time-based billing, vehicle number registration, and GUI integration for a complete real-world simulation.

Source Code:

```
Carparking.java
```

```
/*  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt  
to change this license  
  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit  
this template  
*/
```

```
package carparking;
```

```
import java.util.Scanner;
```

```
/**  
 *  
 * @author Lenovo1  
 */
```

```
public class Carparking {
```

```
/**  
 * @param args the command line arguments  
 */
```

```
public static void main(String[] args) {  
    // TODO code application logic here
```

```
    parking par =new parking();
```

```
Scanner sc=new Scanner(System.in);

int choice;

int car=0,bike=0,rikshow=0;

while(true)

{

    System.out.println("press 1 to enter car");

    System.out.println("press 2 to enter bike");

    System.out.println("press 3 to enter rikshow");

    System.out.println("press 4 to view data");

    System.out.println("press 5 to delete data");

    System.out.println("press 6 to exit");

    choice=sc.nextInt();

    switch (choice)

    {

        case 1:

            car++;

            par.setcar(car);

            System.out.println(car+"cars is added");

            break;

        case 2:

            bike++;

            par.setbike(bike);

            System.out.println(bike+"bikes is added");

            break;

        case 3:

            rikshow++;

            par.setrikshow(rikshow);

            System.out.println(rikshow+"rikshaws is added");

            break;

        case 4:

            System.out.println("total cars "+car);

            System.out.println("total bikes "+bike);

            System.out.println("total rikshaws "+rikshow);

            break;

        case 5:

            car=0;

            bike=0;

            rikshow=0;

            System.out.println("data deleted");

            break;

        case 6:

            System.out.println("exit");

            break;

        default:

            System.out.println("invalid choice");

    }

}
```

```
        par.setbike(bike);

        System.out.println(bike+"bikes is added");

        break;

    case 3:

        bike++;

        par.setrikshow(rikshow);

        System.out.println(rikshow+"rikshows is added");

        break;

    case 4:

        System.out.println("\n cars:"+par.getcar());

        System.out.println("\n bikes:"+par.getbike());

        System.out.println("\n rikshows:\n "+par.getrikshow());



    }

}

}

parking.java

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 * to change this license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Other/File.java to edit this
 * template

```

```
 */
```

```
package carparking;
```

```
/**
```

```
*
```

```
* @author Lenovo
```

```
*/
```

```
public class parking {
```

```
/**
```

```
* @param args the command line arguments
```

```
*/
```

```
private int car,bike,rikshow;
```

```
parking()
```

```
{
```

```
    car=0;
```

```
    bike=0;
```

```
    rikshow=0;
```

```
}
```

```
public int getcar(){
```

```
    return car;
```

```
}
```

```
public void setcar(int car){
```

```
this.car=car;  
}  
  
public int getbike(){  
    return bike;  
}  
  
public void setbike(int bike){  
    this.bike=bike;  
}  
  
public int getrikshow(){  
    return rikshow;  
}  
  
public void setrikshow(int rikshow){  
    this.rikshow=rikshow;  
}  
}
```

Output Screenshots:

```
carparking - Apache NetBeans (IDE 24)
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F) ×
Projects X
carparking
Kharal
Kharal2.0
mavaproject1
Mavaproject1
SealedData
SealedGame
zparking.java
Source History ▾
public class zparking implements ActionListener {
    package zparking;
    import javax.swing.*;
    import java.awt.*;
    import java.awt.event.*;
    ...
    public class Main extends JFrame implements ActionListener {
        ...
        private Image apple;
        private Image dots;
        private Image heads;
        ...
        private final int ALL_ZONE = 600;
        private final int DOT_SIZE = 10;
        private final int RANDOM_POSITION = 29;
        ...
        private int apple_x;
        private int apple_y;
        ...
        private final int x[] = new int[ALL_ZONE];
        private final int y[] = new int[ALL_ZONE];
        ...
        private boolean leftDirection = false;
    }
}
Output: carparking (run) ×
press 1 to enter car
press 2 to enter bike
press 3 to enter vikarsh
press 4 to view data
press 5 to delete data
press 6 to exit
```

Figure 1.1 Menu Display

```
mavaproject1 - Apache NetBeans (IDE 24)
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+F) ×
Projects X
carparking
Kharal
Kharal2.0
mavaproject1
Mavaproject1
SealedData
SealedGame
zparking.java
Source History ▾
public class zparking implements ActionListener {
    package zparking;
    import javax.swing.*;
    import java.awt.*;
    import java.awt.event.*;
    ...
    public class Main extends JFrame implements ActionListener {
        ...
        private Image apple;
        private Image dots;
        private Image heads;
        ...
        private final int ALL_ZONE = 600;
        private final int DOT_SIZE = 10;
        private final int RANDOM_POSITION = 29;
        ...
        private int apple_x;
        private int apple_y;
        ...
        private final int x[] = new int[ALL_ZONE];
        private final int y[] = new int[ALL_ZONE];
        ...
        private boolean leftDirection = false;
    }
}
Output: carparking (run) ×
press 1 to enter car
press 2 to enter bike
press 3 to enter vikarsh
press 4 to view data
press 5 to delete data
press 6 to exit
2
initials is added
press 1 to enter car
press 2 to enter bike
press 3 to enter vikarsh
press 4 to view data
press 5 to delete data
press 6 to exit
```

Figure 1.2 When Vehicles Added

```
1 package carparking;
2
3 import java.awt.*;
4 import java.awt.event.*;
5
6 public class Board extends JPanel implements ActionListener {
7
8     private Image apple;
9     private Image dots;
10    private Image head;
11
12    private final int ALL_ROWS = 600;
13    private final int DOT_SIZE = 10;
14    private final int RANDOM_POSITION = 20;
15
16    private int apple_x;
17    private int apple_y;
18
19    private final int X[] = new int[ALL_ROWS];
```

Output - carparking (run) :

```
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
1
Bikes is added
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
```

Figure 1.3 bikes added

```
1 package carparking;
2
3 import java.awt.*;
4 import java.awt.event.*;
5
6 public class Board extends JPanel implements ActionListener {
7
8     private Image apple;
9     private Image dots;
10    private Image head;
11
12    private final int ALL_ROWS = 600;
13    private final int DOT_SIZE = 10;
14    private final int RANDOM_POSITION = 20;
15
16    private int apple_x;
17    private int apple_y;
18
19    private final int X[] = new int[ALL_ROWS];
```

Output - carparking (run) :

```
press 1 to view data
press 2 to delete data
press 6 to exit
3
Bikeshop is added
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
4
carail
bikeshop
rikshaw
0
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
5
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
```

Figure 1.4 View parking

The screenshot shows the NetBeans IDE interface with the 'Output' window open. The 'Output' tab is selected, displaying the following text:

```
Output - parking [out] X Board.java X StateDelete.java X
press 1 to enter data
press 2 to exit
3
0
Bikes is added
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
2
Bikes is added
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
2
Bikes is added
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
4
cancel
bikesis5
rikshawis
0
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
```

Figure 1.5 More Parking Added

The screenshot shows the NetBeans IDE interface with the 'Output' window open. The 'Output' tab is selected, displaying the following text:

```
Output - parking [out] X Board.java X StateDelete.java X
press 1 to enter data
press 2 to exit
3
0
Bikes is added
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
2
Bikes is added
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
2
Bikes is added
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
2
Bikes is added
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
2
Bikes is added
press 1 to enter car
press 2 to enter bike
press 3 to enter rikshaw
press 4 to view data
press 5 to delete data
press 6 to exit
```

Figure 1.6 Exit