DEPARTMENT OF COMPUTER SCIENCE

CSCL2102 – Data Structures and Algorithms

PROJECT REPORT

SHOPPING CART MANAGEMENT

GROUP MEMBERS:

1. HOORIA ALI (2312319)
2. HUMAIRA BIBI (2312420)

Submitted to:

Sir Syed Muhammad Hassan

Lab Engineer Sir Waseem Rauf

Table of Content:

Objective

The objective of this project is to develop a shopping cart system using data structures and algorithms in Java. This system supports basic functionalities such as adding, removing, and replacing items, as well as advanced features like sorting by price or name, searching for items, and calculating total cost including tax. The project includes implementation of stack and array-based dynamic capacity as well as sorting features.

Description

This project implements a shopping cart system as part of a Data Structures and Algorithms (DSA) application. The system supports operations like dynamically growing the cart's capacity, sorting items, and calculating total costs with applied taxes. The project utilizes a stack for internal operations and incorporates sorting algorithms like Quicksort to manage the cart's contents effectively.

Concepts Implemented

Object-Oriented Programming (OOP):
The classes used encapsulation functionalities so the user will not know about the inner working or data of the project.

The shopping cart and Item classes use constructors to initialize objects with specific attributes.

The item class also implements method overriding that provides a string representation of the object.

The project uses private attributes in classes to ensure data hiding, with public methods to access and modify these attributes.

Stack Data Structure:

Using for managing cart operations basically in adding items in cart.

Dynamic Arrays:

Provides dynamic resizing of the cart when the cart is full it automatically expands by multiple of the capacity.

Sorting Algorithms:

Quicksort implementation for sorting items by price or name.

Tax Calculation:

Total amount to be paid by calculating tax based on individual item rates.

Search Functionality:

Efficient searching for items by name to check if that particular item is already in the cart or not.

Project Features

Add Item to Cart:

It dynamically adds items to the cart.

It also resizes cart if the initial capacity is exceeded by extending by multiple of capacity.

Remove Item from Cart:

Removes an item from a specified zero-base indexing entered by user.

Replace Item in Cart:

Replaces an item with a new one at a specified index entered by user.

View Cart:

Displays the details of all items in the cart.

Search Item:

Searches for an item by name and returns its details if found.

Calculate Total Tax:

Computes the total price and total price including tax for all items in the cart.

Sort Items:

Quicksort implementation to sort items by price or name so the user gets an idea about the prices of their selected items.

Dynamic Capacity Management:

Automatically grows the cart size when it reaches capacity.

Source Code

ITEM CLASS:

```java
package PROJECTOFDSA;

public class Item {
      String Name;
        double price;
        String Category;
        boolean isFavourite;
        double taxRate;



        public Item(String Name, double price, String category, boolean isFavourite ,
double taxRate) {
            this.Name = Name;
            this.price = price;
            this.Category = category;
            this.isFavourite = isFavourite;
            this.taxRate = taxRate;

        }



    public double calculateTax() {
        return (this.price * this.taxRate) / 100; }
        @Override
        public String toString() {
            return "Item{" +
                    "Name='" + Name + '\'' +
                    ", price=" + price +
                    ", Category='" + Category + '\'' +
                    ", isFavourite=" + isFavourite +
                    '}';
        }
    }
```

STACK CLASS:

```java
package PROJECTOFDSA;
```

```
 public class Stack {

    private Item[] items;
    private int top;

    public Stack(int Capacity) {
     items = new Item[Capacity];
        top = -1;
    }

     public void push(Item item) {
         if (top < items.length - 1) {
             items[++top] = item;
         } else {
             System.out.println("Stack overflow");
         }
     }

}
```

SHOPPING CART CLASS:

```
package PROJECTOFDSA;


public class ShoppingCart {

    Stack stack;
  Item[] Cart;
   int Size;
    int Capacity;
    public ShoppingCart(int Initial_Capacity) {
    this.Capacity = Initial_Capacity;
    this.Cart = new Item[Capacity];
    this.stack = new Stack(Capacity);
    this.Size = 0;
    System.out.println();
    System.out.println("\n---WELCOME TO SHOPPING CART---\n");
    System.out.print(" Shopping cart is initializes with capacity: " + Capacity)
    private void Grow() {
        int newCapacity = Capacity * 2;
        Item[] newCart = new Item[newCapacity];
        for (int i = 0; i < Size; i++) {
            newCart[i] = Cart[i];
        }
        Cart = newCart;
        Capacity = newCapacity;
        System.out.println("Cart capacity is extended to : " + Capacity);
    }

    public void AddItem(Item item) {
        if (Size >= Capacity) {
            Grow();
        }
        stack.push(item);
        Cart[Size] = item;
        Size++;
        System.out.println(item.Name + "has been added to the cart ");
    }



      public void removeItem(int index) {
            if (index < 0 || index >= Size)
                throw new IndexOutOfBoundsException("Invalid index.Cart is Empty");
            for(int i= index;i <Size-1;i++)  {
                Cart[i] = Cart[i + 1];
            }
            Cart[Size - 1] = null;
            Size--;
            System.out.println(" Item at index " + index + " has been removed from the
cart.");
        }



        public void ViewCart() {
            if (Size == 0) {
                System.out.println("Cart is empty.Nothing to view");
```

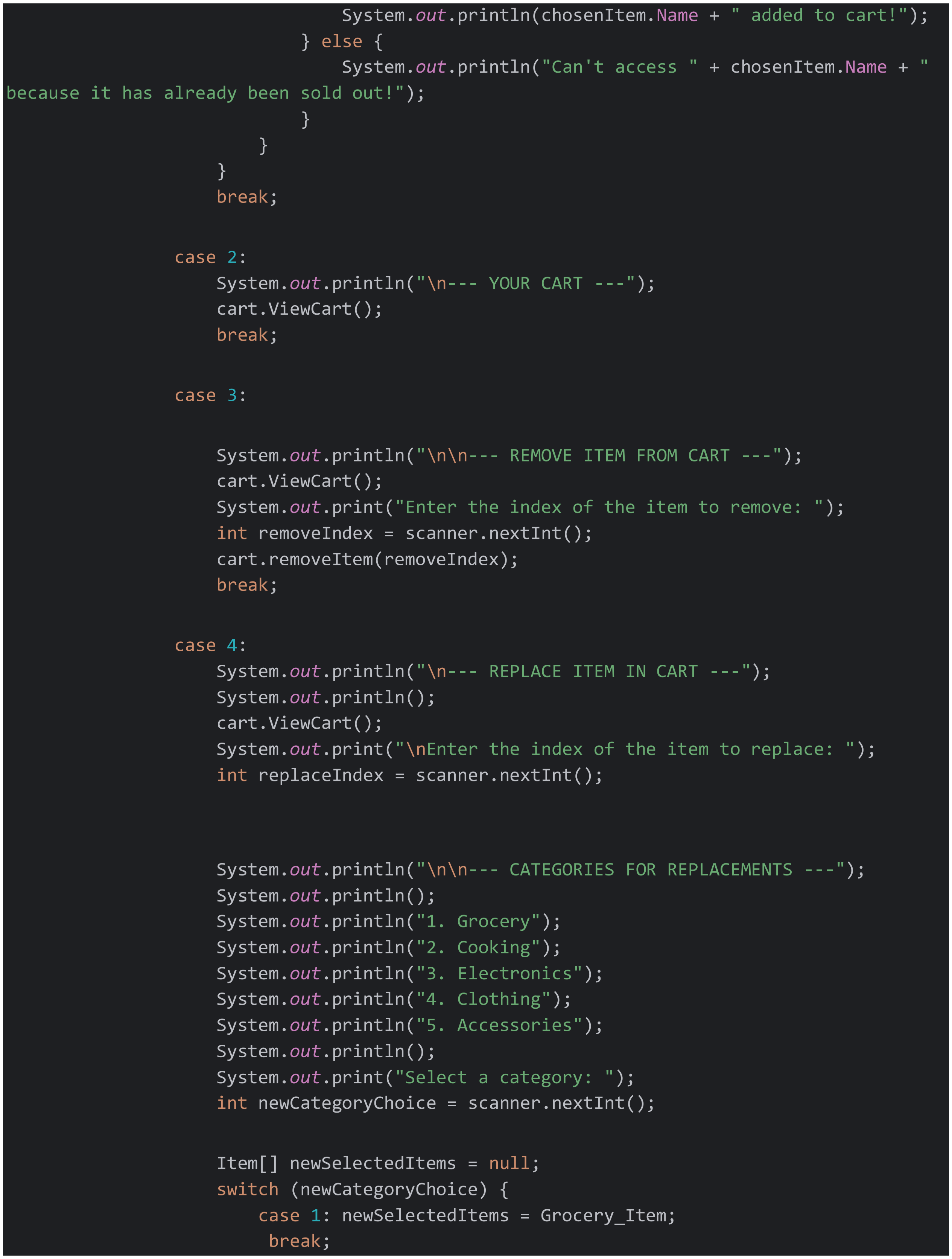
```java
            return;
        } else {
            System.out.println("Item in the cart: \n");
            for (int i = 0; i < Size; i++) {
                System.out.println(i + " : " + Cart[i]);
            }
        }
    }



    public void QuickSortByPrice() {
        QuickSort(Cart,0,Size- 1,"Price");
        System.out.println("Cart sorted by price using Quick Sort");
    }
    public void QuickSortByName() {
        QuickSort(Cart,0,Size-1,"Name");
        System.out.println("Cart sorted alphabetically using Quick Sort");
    }

    private void QuickSort(Item[] arr, int low, int high, String sortBy) {
        if (low < high) {
            int pi = partition(arr, low, high, sortBy);
            QuickSort(arr, low, pi - 1, sortBy);
            QuickSort(arr, pi + 1, high, sortBy);
        }
    }

    private int partition(Item[] arr, int low, int high, String sortBy) {
        Item pivot = arr[high];
        int i = (low - 1);
        for (int j = low; j < high; j++) {
            boolean condition;
            if (sortBy.equals("Price")) {
                condition = arr[j].price <= pivot.price;
            } else {
                condition = arr[j].Name.compareToIgnoreCase(pivot.Name) <= 0;
            }
            if (condition) {
                i++;
                Item temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
        Item temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;
```

```
            return i + 1;
        }



        public void ReplacingItem(int index, Item newitem){
            if(index<0 || index>=Size){
                System.out.println("Invalid index! Can't replace item");
                return;
            }
            Item Olditem = Cart[index];
            Cart[index]=newitem;
            System.out.println(" Index "  +  index  + " has been replaced  with : " +
newitem);
        }

        public double CalculateTotalTax() {
        double total = 0;
        double totalWithTax = 0;

        for (int i = 0; i < Size; i++) {
            Item item = Cart[i];
            total += item.price;
            double itemTax = item.calculateTax();
            totalWithTax += item.price + itemTax;
        }
        System.out.println("Total: Rs." + total + ", Total with tax: Rs." +
totalWithTax);
        return totalWithTax;
    }



    public boolean SearchItem(String Name) {
        boolean found = false;
        for (int i=0; i<Size;i++) {
            if (Cart[i].Name.equalsIgnoreCase(Name)) {
                System.out.println("Found: " + Cart[i]);
                found=true;
                break;
            }
        }
        if (!found) {
            System.out.println("Item '" + Name + "' not found in the cart.");
        }
        return found;
    }
```

```
}
```

MAIN CLASS:

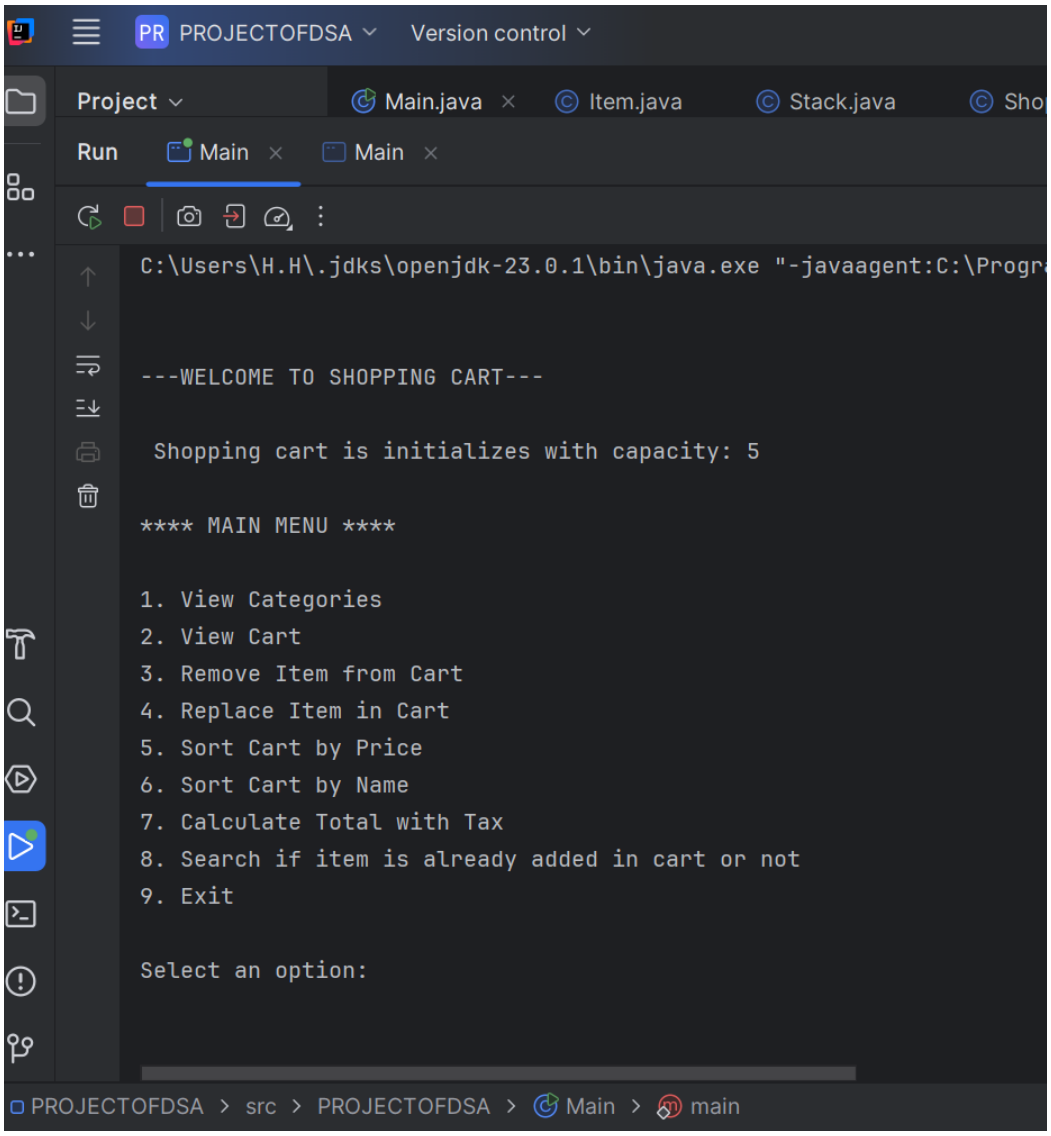
```java
package PROJECTOFDSA;

import java.util.Scanner;


public class Main {
 public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        ShoppingCart cart = new ShoppingCart(5);

        double Grocery_Tax= 5.25;
        double Cooking_Tax= 8.95;
        double Electronic_Tax= 10.98;
        double Clothing_Tax= 7.99;
        double Accessories_Tax= 6.33;



        Item[] Grocery_Item = {
                new Item("Rice", 120.0,"Grocery",true,Grocery_Tax),
                new Item("Wheat",200.1,"Grocery",true,Grocery_Tax),
                new Item("Sugar",90.0,"Grocery",false,Grocery_Tax),
                new Item("Oil",400.0,"Grocery",true,Grocery_Tax),
                new Item("Ketchup",250.0,"Grocery",true,Grocery_Tax)
        };

        Item[] Cooking_Item = {
                new Item("Salt",100.0,"Cooking",true,Cooking_Tax),
                new Item("Vinegar",250.0,"Cooking",true,Cooking_Tax),
                new Item("Pepper",150.0,"Cooking",false,Cooking_Tax),
                new Item("Garlic",200.5,"Cooking",true,Cooking_Tax),
                new Item("Ginger",300.9,"Cooking",true,Cooking_Tax)
        };

        Item[] Electronics_Items = {
                new Item("Laptop",50000.0,"Electronics",true,Electronic_Tax),
                new Item("Mobile phone",30000.3,"Electronics",true,Electronic_Tax),
                new Item("Headphones",2000.99,"Electronics",false,Electronic_Tax),
                new Item("Smart watch",15000.09,"Electronics",true,Electronic_Tax),
                new Item("Camera",100000.10,"Electronics",true,Electronic_Tax)
        };

        Item[] Clothing_Items = {
                new Item("T-Shirt",500.2,"Clothing",true,Clothing_Tax),
                new Item("Jeans",1500.3,"Clothing",true,Clothing_Tax),
                new Item("Jacket",3000.1,"Clothing",false,Clothing_Tax),
                new Item("Shoes",2000.8,"Clothing",true,Clothing_Tax),
                new Item("Hat",400.1,"Clothing",true,Clothing_Tax)
        };

        Item[] Accessories_Items = {
                new Item("Watch",2502.9,"Accessories",true,Accessories_Tax),
                new Item("Sunglasses",1503.5,"Accessories",true,Accessories_Tax),
                new Item("Earrings",1000.5,"Accessories",false,Accessories_Tax),
                new Item("Ring",1200.9,"Accessories",true,Accessories_Tax),
                new Item("Necklace",2050.10,"Accessories",true,Accessories_Tax)
        };


        while (true) {

            System.out.println();
            System.out.println("\n**** MAIN MENU ****");
            System.out.println();
            System.out.println("1. View Categories");
            System.out.println("2. View Cart");
            System.out.println("3. Remove Item from Cart");
            System.out.println("4. Replace Item in Cart");
            System.out.println("5. Sort Cart by Price");
            System.out.println("6. Sort Cart by Name");
            System.out.println("7. Calculate Total with Tax");
            System.out.println("8. Search if item is already added in cart or not");
            System.out.println("9. Exit");
            System.out.println();
            System.out.print("Select an option: ");

            int selection = scanner.nextInt();

            switch (selection) {
                case 1:
                    System.out.println();
                    System.out.println("\n--- CATEGORIES ---");
                    System.out.println("1. Grocery");
                    System.out.println("2. Cooking");
                    System.out.println("3. Electronics");
                    System.out.println("4. Clothing");
                    System.out.println("5. Accessories");
                    System.out.println();
                    System.out.print  ("Select a category: ");
                    int categoryChoice = scanner.nextInt();

                    Item[] selectedItem = null;

                    switch (categoryChoice) {

                        case 1: selectedItem = Grocery_Item;
                          break;
                        case 2: selectedItem = Cooking_Item;
                          break;
                        case 3: selectedItem = Electronics_Items;
                          break;
                        case 4: selectedItem = Clothing_Items;
                           break;
                        case 5: selectedItem = Accessories_Items;
                          break;
                        default: System.out.println("Invalid category choice!");
                           break;
                    }

                    if(selectedItem != null) {
                        System.out.println();
                        System.out.println("\n--- ITEMS IN SELECTED CATEGORY ---");

                        for(int i = 0; i < selectedItem.length; i++) {
                            Item item = selectedItem[i];
                            String availability = item.isFavourite ? "Available" : "Sold
Out";

                            System.out.println(i + 1 + ". " + item.Name + " - Rs." +
item.price + " (" + availability + ")");
                        }



                        System.out.println();
                        System.out.print("Enter the item number to add to cart or 0 to go
back: ");
                        int itemChoice = scanner.nextInt();
                        if (itemChoice > 0 && itemChoice <= selectedItem.length) {
                            Item chosenItem = selectedItem[itemChoice - 1];
                            if(chosenItem.isFavourite) {
                                cart.AddItem(chosenItem);
```



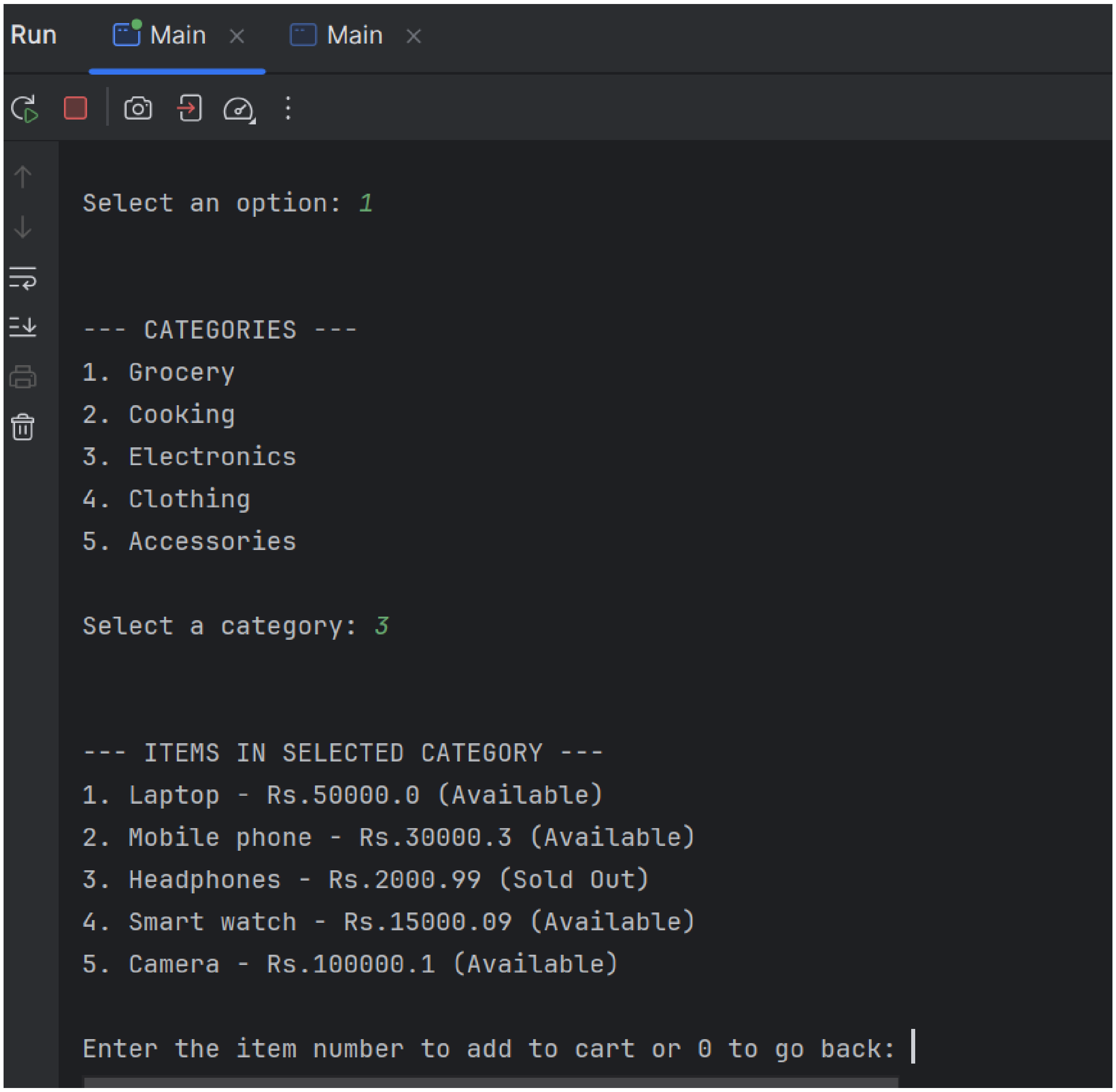
```
                                System.out.println(chosenItem.Name + " added to cart!");
                            } else {
                                System.out.println("Can't access " + chosenItem.Name + "
because it has already been sold out!");
                            }
                        }
                    }
                    break;

                case 2:
                    System.out.println("\n--- YOUR CART ---");
                    cart.ViewCart();
                    break;

                case 3:

                    System.out.println("\n\n--- REMOVE ITEM FROM CART ---");
                    cart.ViewCart();
                    System.out.print("Enter the index of the item to remove: ");
                    int removeIndex = scanner.nextInt();
                    cart.removeItem(removeIndex);
                    break;

                case 4:
                    System.out.println("\n--- REPLACE ITEM IN CART ---");
                    System.out.println();
                    cart.ViewCart();
                    System.out.print("\nEnter the index of the item to replace: ");
                    int replaceIndex = scanner.nextInt();



                    System.out.println("\n\n--- CATEGORIES FOR REPLACEMENTS ---");
                    System.out.println();
                    System.out.println("1. Grocery");
                    System.out.println("2. Cooking");
                    System.out.println("3. Electronics");
                    System.out.println("4. Clothing");
                    System.out.println("5. Accessories");
                    System.out.println();
                    System.out.print("Select a category: ");
                    int newCategoryChoice = scanner.nextInt();

                    Item[] newSelectedItems = null;
                    switch (newCategoryChoice) {
                        case 1: newSelectedItems = Grocery_Item;
                         break;
                        case 2: newSelectedItems = Cooking_Item;
                         break;
                        case 3: newSelectedItems = Electronics_Items;
                        break;
                        case 4: newSelectedItems = Clothing_Items;
                         break;
                        case 5: newSelectedItems = Accessories_Items;
                        break;
                        default: System.out.println("Invalid category choice!");
                        break;
                    }

                    if (newSelectedItems != null) {
                        System.out.println();
                        System.out.println("\n--- ITEMS IN SELECTED CATEGORIES ---");
                           for (int i = 0; i < newSelectedItems.length; i++) {
                              Item item = newSelectedItems[i];
                             String availability = item.isFavourite ? "Available" : "Sold
Out";

                             System.out.println(i + 1 + ". " + item.Name + " - Rs." +
item.price + " (" + availability + ")");
                        }

                        System.out.println();
                        System.out.print("Enter the item number to replace with: ");
                        int newItemChoice = scanner.nextInt();

                        if (newItemChoice > 0 && newItemChoice <=
newSelectedItems.length) {
                            Item newChosenItem = newSelectedItems[newItemChoice - 1];
                            if ( newChosenItem.isFavourite ) {
                                  cart.ReplacingItem(  replaceIndex,  newChosenItem);
                                 System.out.println();
                                 System.out.println( newChosenItem.Name  +  " has been
replaced in the cart!");
                            } else {
                                System.out.println("Can't access " + newChosenItem.Name +
" because it has already been sold out!");
                            }
                        }
                    }
                    break;

                    case 5:
                    System.out.println("\n--- SORTING CART BY PRICE ---");
                     cart.QuickSortByPrice();
                    cart.ViewCart();
```

```java
                    break;

                    case 6:
                    System.out.println("\n--- SORTING CART BY NAME ---");
                     cart.QuickSortByName();
                     cart.ViewCart();
                    break;

                    case 7:
                    System.out.println("\n--- TAX RATES PER CATEGORY --- ");
                    System.out.println("Tax for grocery items is : " + Grocery_Tax);
                    System.out.println("Tax for cooking items is : " + Clothing_Tax);
                    System.out.println("Tax for electronic items is : " +
Electronic_Tax);

                    System.out.println("Tax for clothing items is : " + Clothing_Tax);
                    System.out.println("Tax for accessories items is : " +
Accessories_Tax);

                    System.out.println("\n--- Calculating Total with Tax ---");
                     cart.CalculateTotalTax();
                     break;



                case 8:
                    System.out.print("Enter the name of the item to search: ");
                    scanner.nextLine();
                    String searchName = scanner.nextLine();
                    cart.SearchItem(searchName);
                    break;

                case  9:
                     System.out.println("EXITING PROGRAM.PLEASE COME AGAIN !");
                     scanner.close();
                    return;

                default:
                    System.out.println("Invalid choice! Please try again.");
            }
        }
    }
}
```

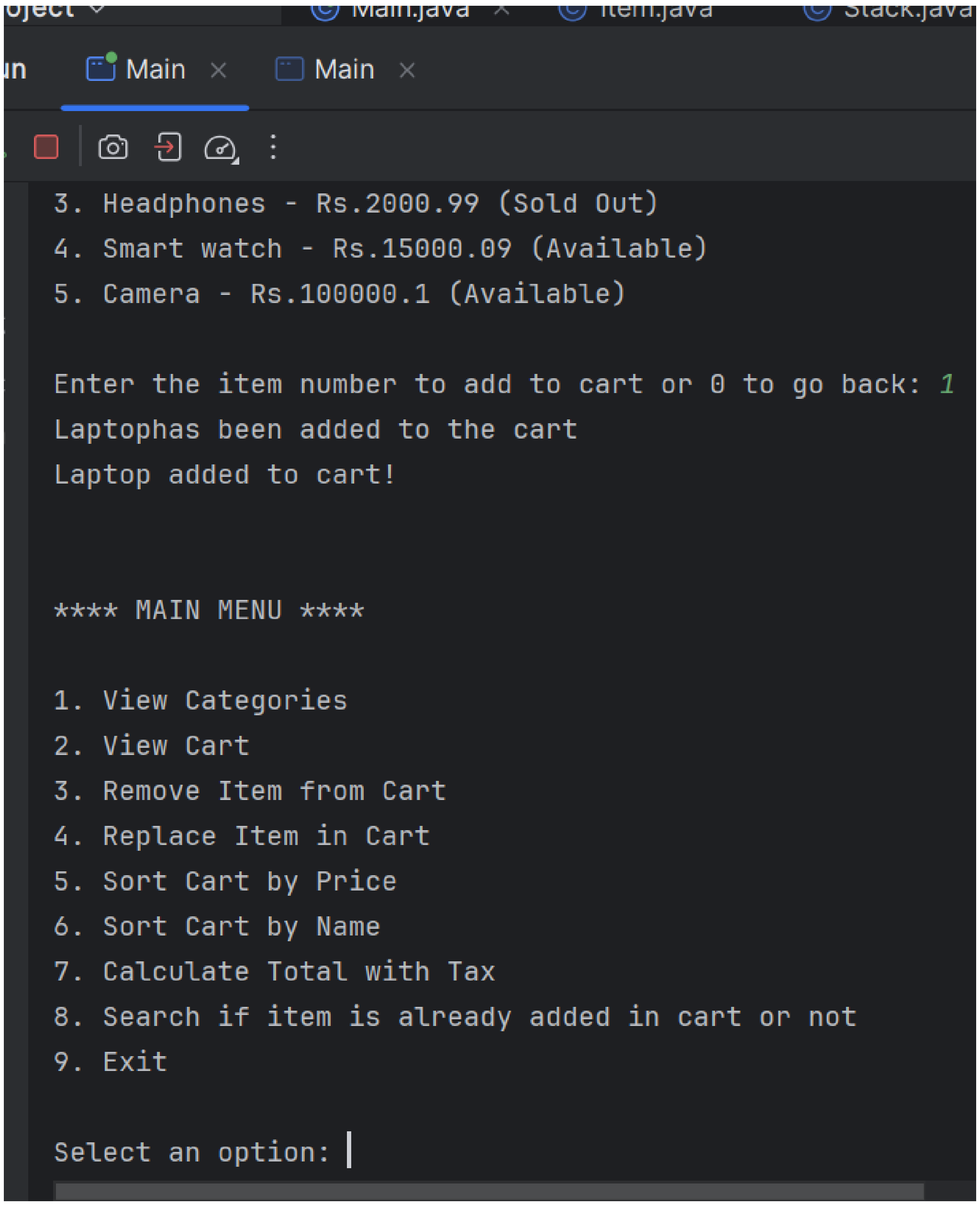
Output



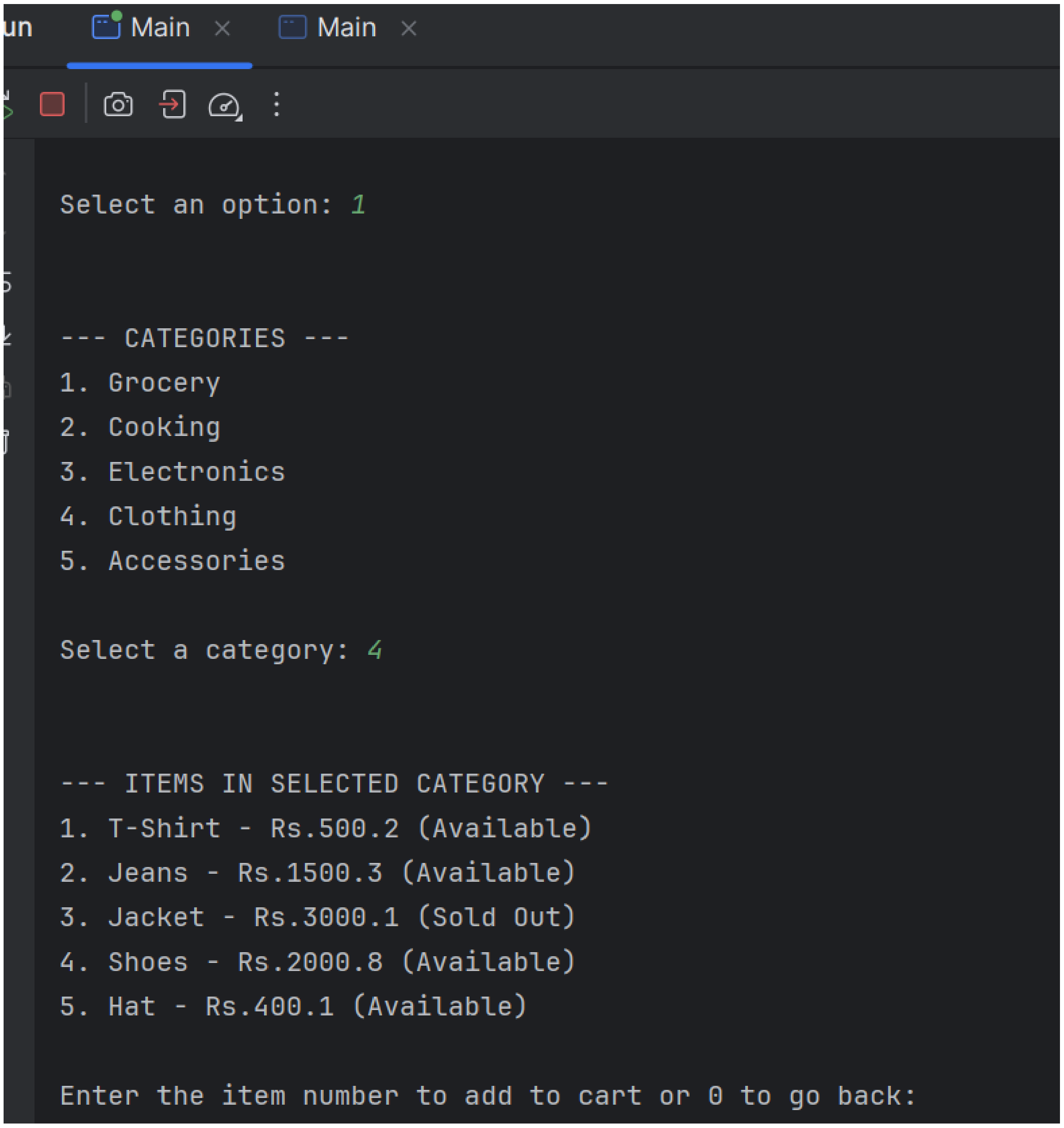
```
C:\Users\H.H\.jdks\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Progr


---WELCOME TO SHOPPING CART---

 Shopping cart is initializes with capacity: 5

**** MAIN MENU ****

1. View Categories
2. View Cart
3. Remove Item from Cart
4. Replace Item in Cart
5. Sort Cart by Price
6. Sort Cart by Name
7. Calculate Total with Tax
8. Search if item is already added in cart or not
9. Exit

Select an option:
```



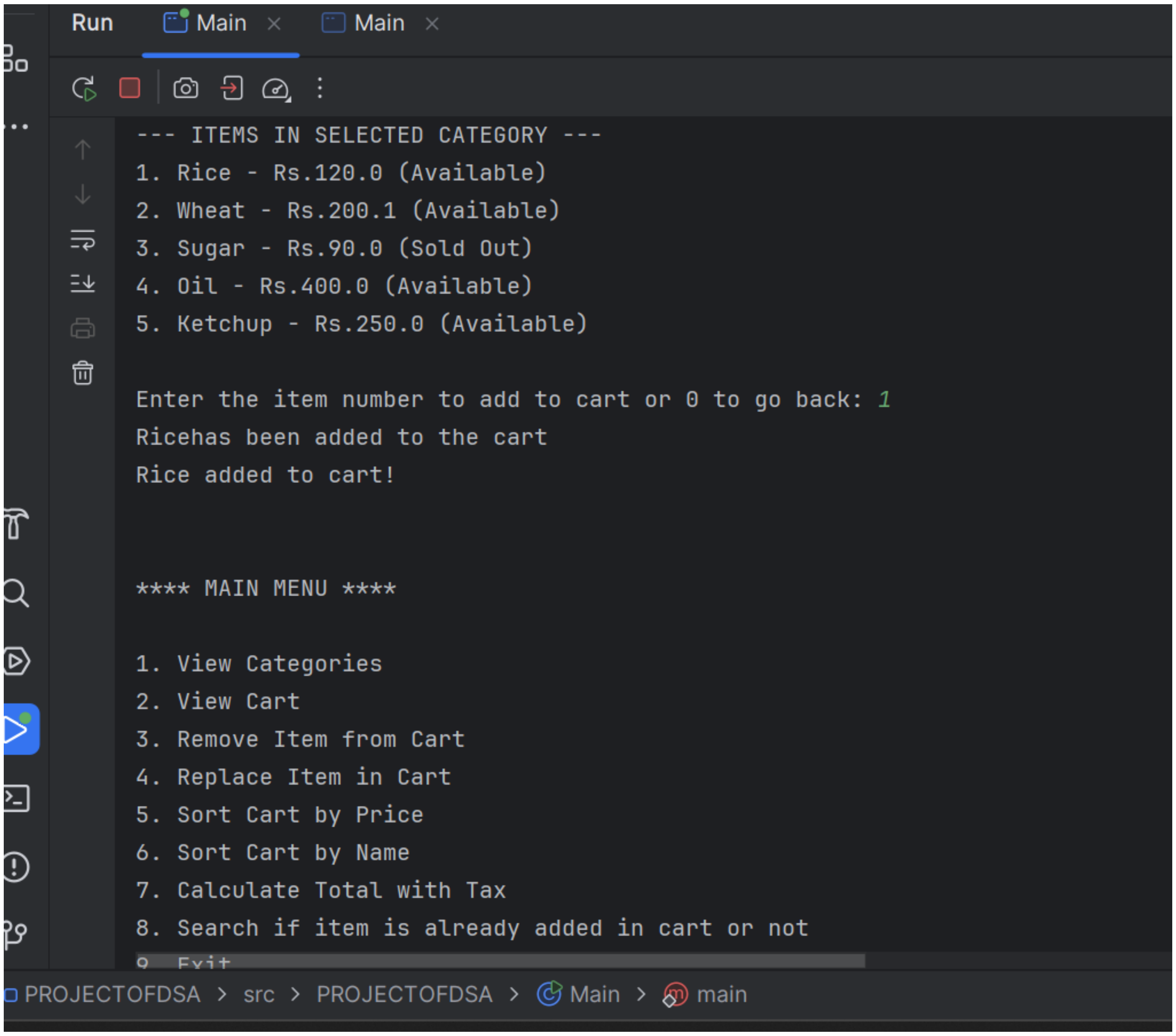
```
Select an option: 1


--- CATEGORIES ---
1. Grocery
2. Cooking
3. Electronics
4. Clothing
5. Accessories

Select a category: 3


--- ITEMS IN SELECTED CATEGORY ---
1. Laptop - Rs.50000.0 (Available)
2. Mobile phone - Rs.30000.3 (Available)
3. Headphones - Rs.2000.99 (Sold Out)
4. Smart watch - Rs.15000.09 (Available)
5. Camera - Rs.100000.1 (Available)

Enter the item number to add to cart or 0 to go back:
```



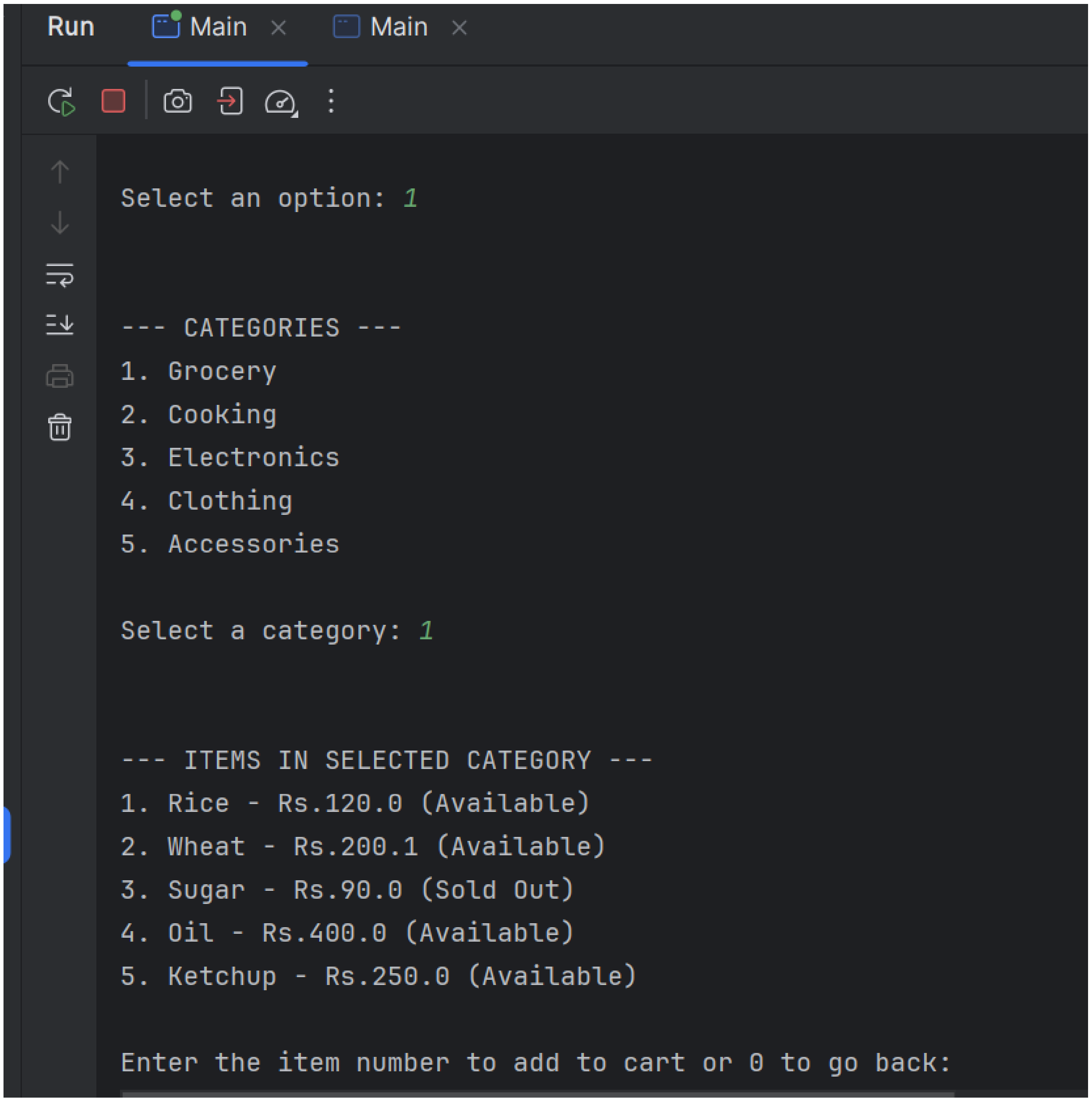
```
3. Headphones - Rs.2000.99 (Sold Out)
4. Smart watch - Rs.15000.09 (Available)
5. Camera - Rs.100000.1 (Available)

Enter the item number to add to cart or 0 to go back: 1
Laptophas been added to the cart
Laptop added to cart!



**** MAIN MENU ****

1. View Categories
2. View Cart
3. Remove Item from Cart
4. Replace Item in Cart
5. Sort Cart by Price
6. Sort Cart by Name
7. Calculate Total with Tax
8. Search if item is already added in cart or not
9. Exit

Select an option: 
```



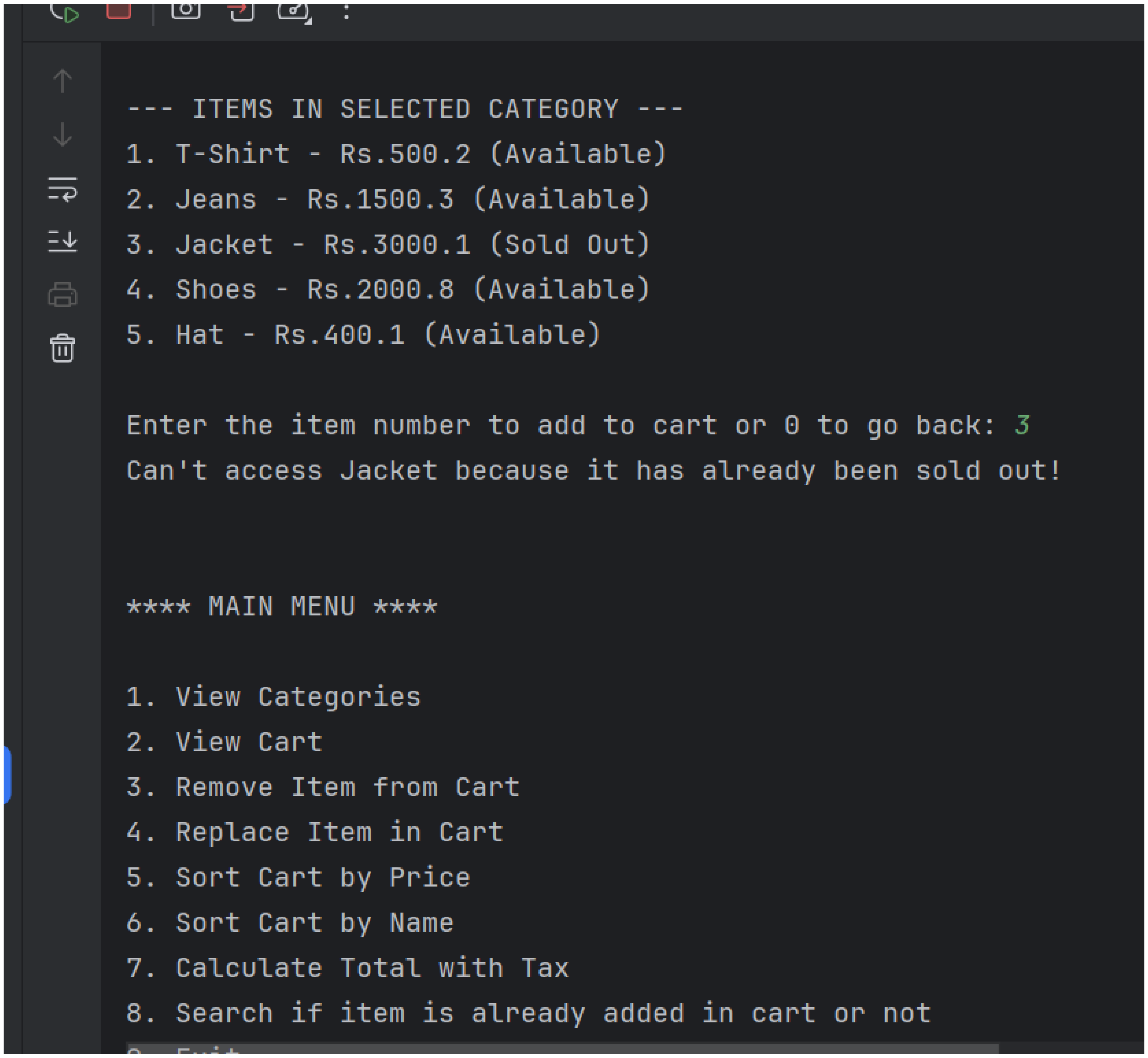
```
Select an option: 1


--- CATEGORIES ---
1. Grocery
2. Cooking
3. Electronics
4. Clothing
5. Accessories

Select a category: 4


--- ITEMS IN SELECTED CATEGORY ---
1. T-Shirt - Rs.500.2 (Available)
2. Jeans - Rs.1500.3 (Available)
3. Jacket - Rs.3000.1 (Sold Out)
4. Shoes - Rs.2000.8 (Available)
5. Hat - Rs.400.1 (Available)

Enter the item number to add to cart or 0 to go back:
```



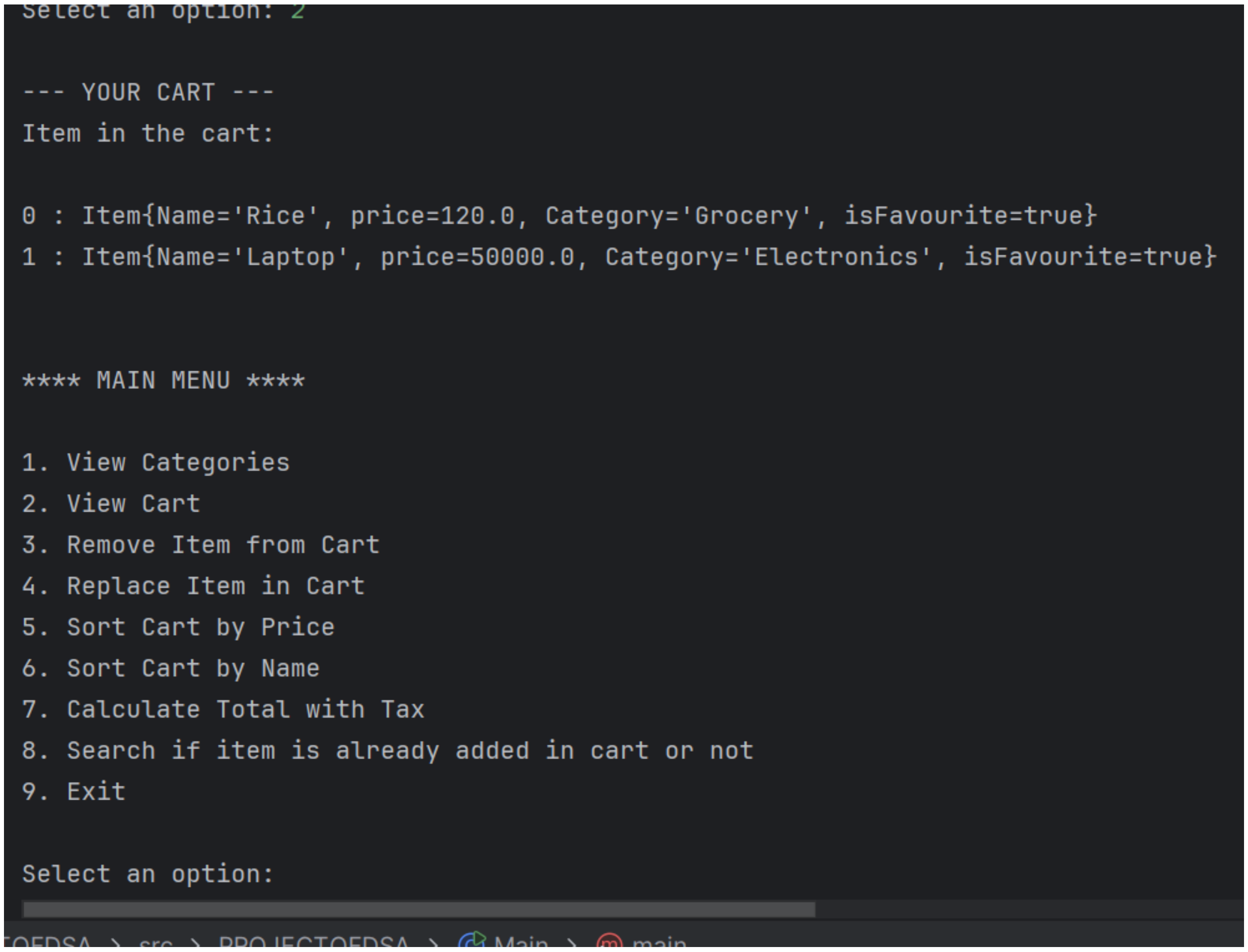
```
--- ITEMS IN SELECTED CATEGORY ---
1. Rice - Rs.120.0 (Available)
2. Wheat - Rs.200.1 (Available)
3. Sugar - Rs.90.0 (Sold Out)
4. Oil - Rs.400.0 (Available)
5. Ketchup - Rs.250.0 (Available)

Enter the item number to add to cart or 0 to go back: 1
Ricehas been added to the cart
Rice added to cart!



**** MAIN MENU ****

1. View Categories
2. View Cart
3. Remove Item from Cart
4. Replace Item in Cart
5. Sort Cart by Price
6. Sort Cart by Name
7. Calculate Total with Tax
8. Search if item is already added in cart or not
9. Exit
```



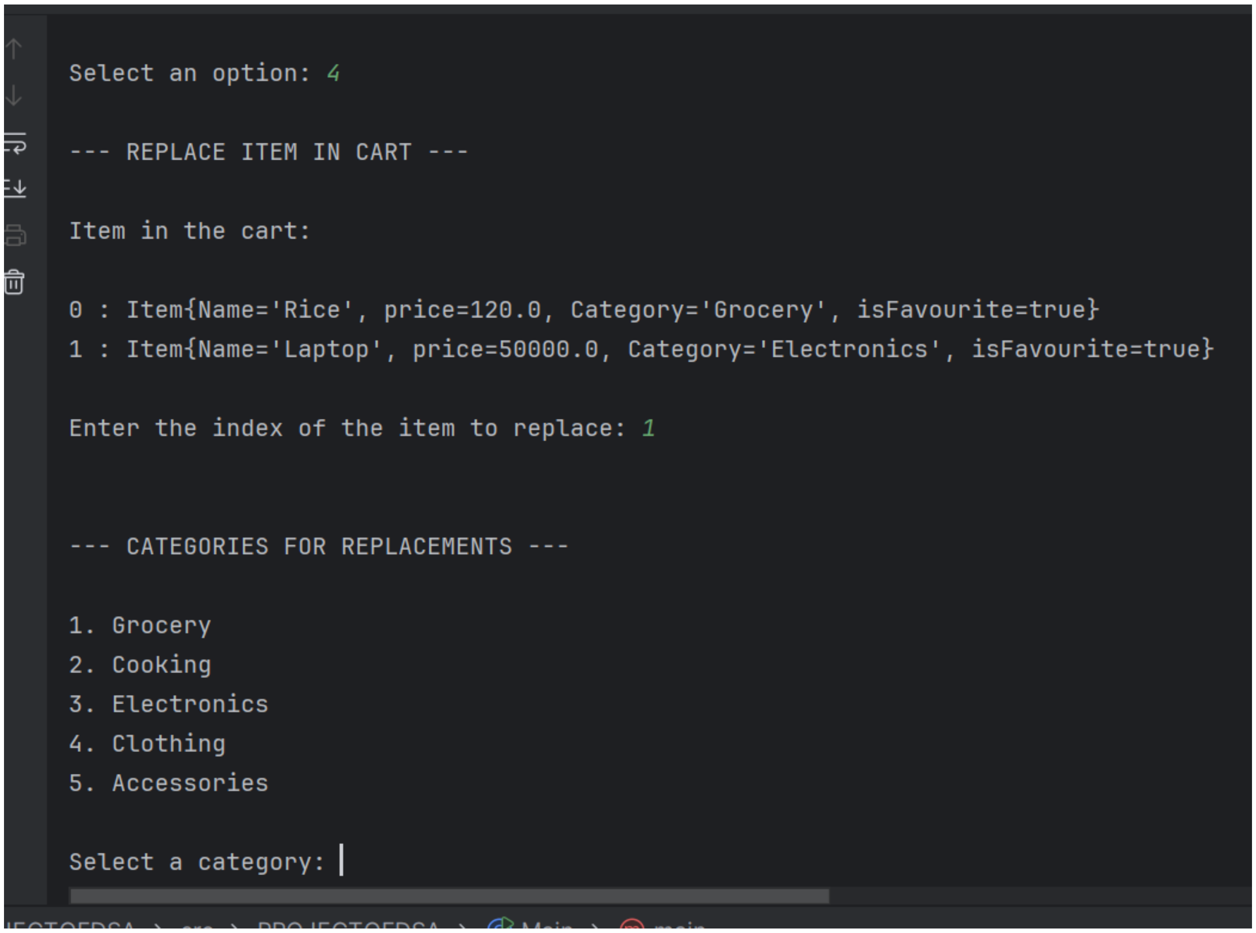
```
Select an option: 1


--- CATEGORIES ---
1. Grocery
2. Cooking
3. Electronics
4. Clothing
5. Accessories

Select a category: 1


--- ITEMS IN SELECTED CATEGORY ---
1. Rice - Rs.120.0 (Available)
2. Wheat - Rs.200.1 (Available)
3. Sugar - Rs.90.0 (Sold Out)
4. Oil - Rs.400.0 (Available)
5. Ketchup - Rs.250.0 (Available)

Enter the item number to add to cart or 0 to go back:
```



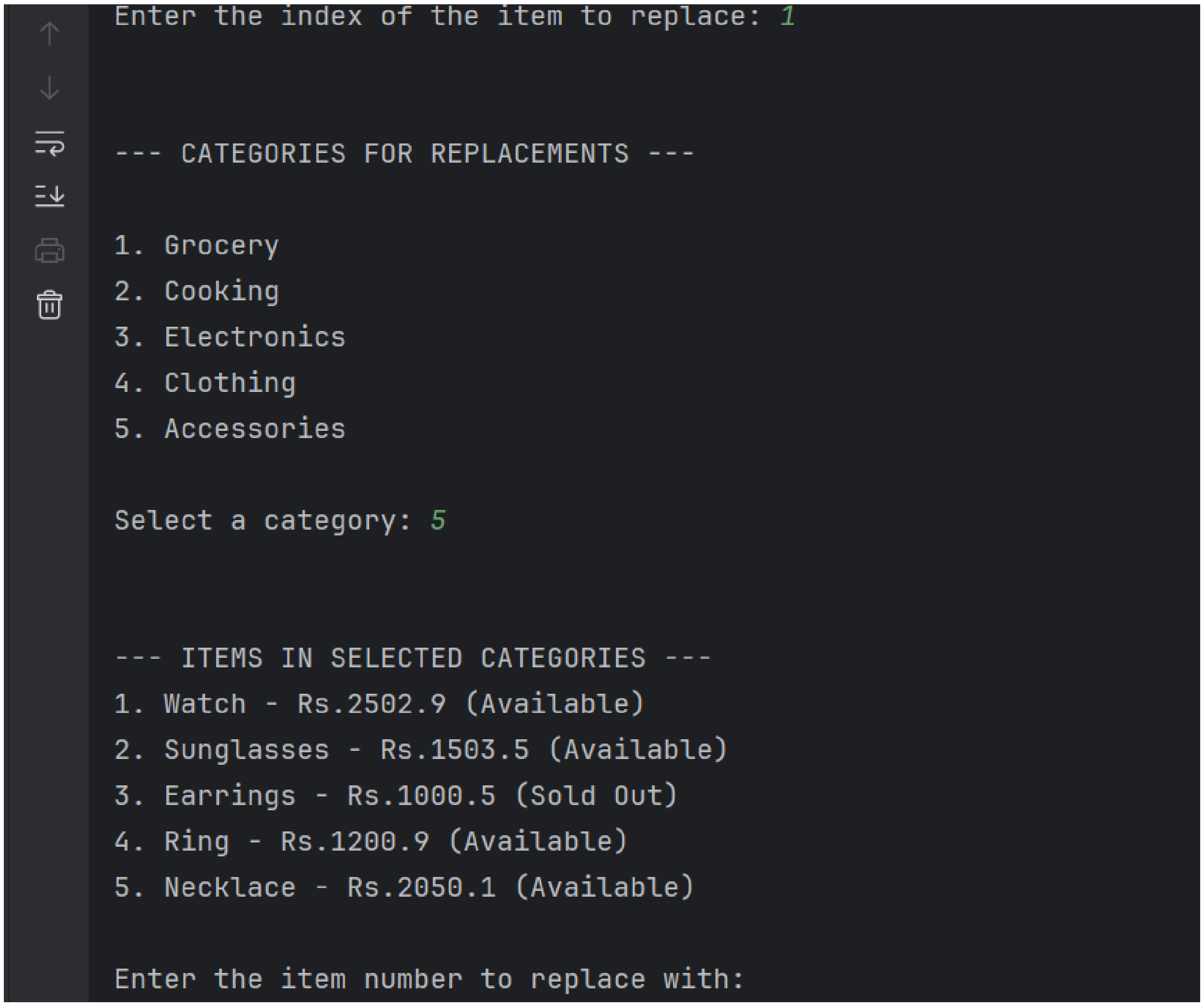
```
--- ITEMS IN SELECTED CATEGORY ---
1. T-Shirt - Rs.500.2 (Available)
2. Jeans - Rs.1500.3 (Available)
3. Jacket - Rs.3000.1 (Sold Out)
4. Shoes - Rs.2000.8 (Available)
5. Hat - Rs.400.1 (Available)

Enter the item number to add to cart or 0 to go back: 3
Can't access Jacket because it has already been sold out!



**** MAIN MENU ****

1. View Categories
2. View Cart
3. Remove Item from Cart
4. Replace Item in Cart
5. Sort Cart by Price
6. Sort Cart by Name
7. Calculate Total with Tax
8. Search if item is already added in cart or not
```



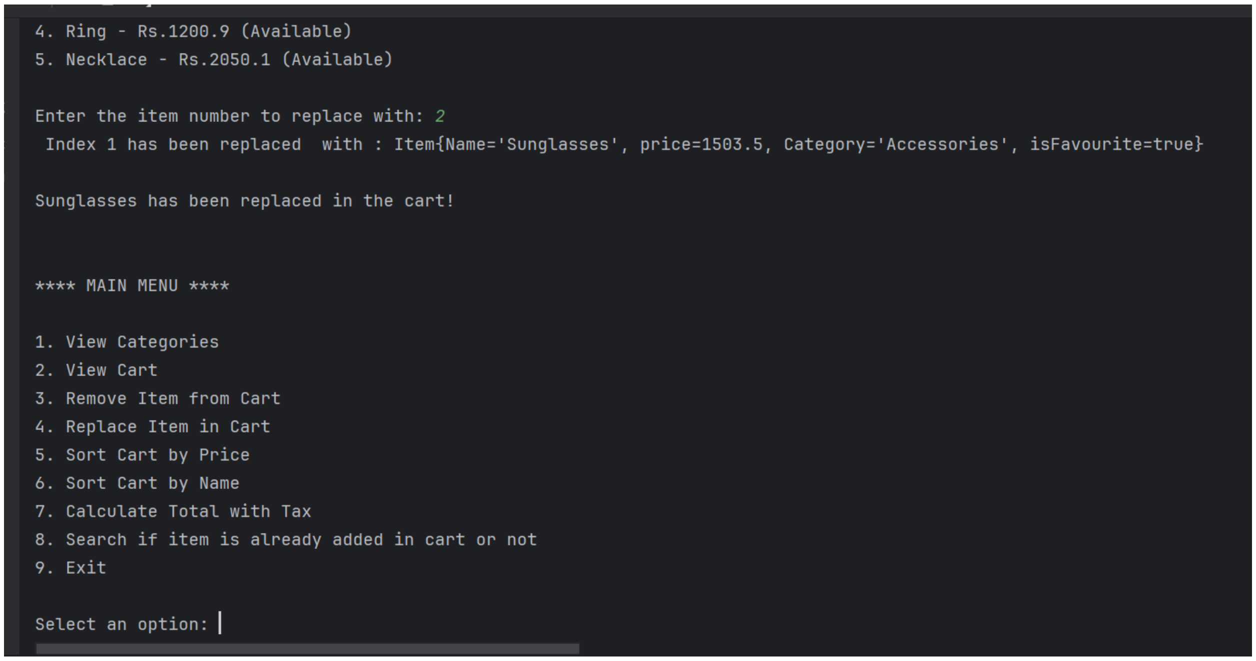
```
Select an option: 2

--- YOUR CART ---
Item in the cart:

0 : Item{Name='Rice', price=120.0, Category='Grocery', isFavourite=true}
1 : Item{Name='Laptop', price=50000.0, Category='Electronics', isFavourite=true}


**** MAIN MENU ****

1. View Categories
2. View Cart
3. Remove Item from Cart
4. Replace Item in Cart
5. Sort Cart by Price
6. Sort Cart by Name
7. Calculate Total with Tax
8. Search if item is already added in cart or not
9. Exit

Select an option:
```



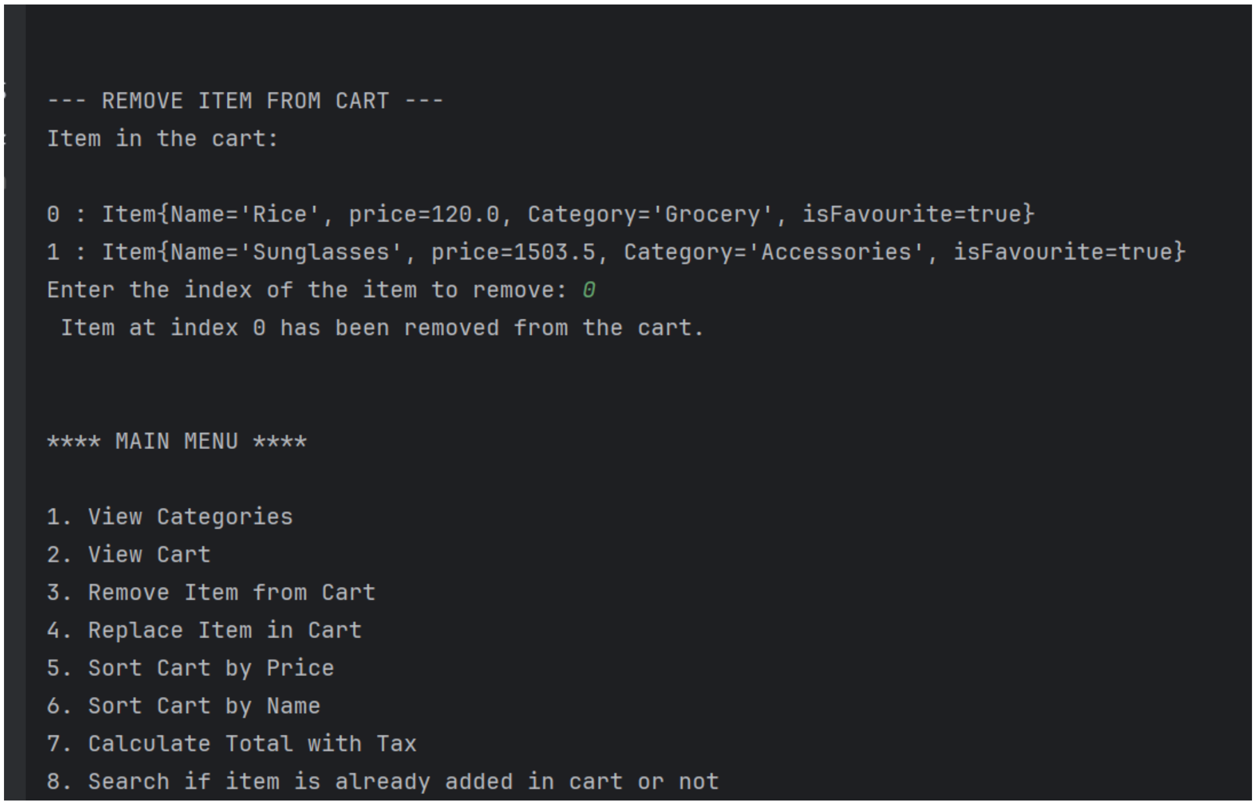
```
Select an option: 4

--- REPLACE ITEM IN CART ---

Item in the cart:

0 : Item{Name='Rice', price=120.0, Category='Grocery', isFavourite=true}
1 : Item{Name='Laptop', price=50000.0, Category='Electronics', isFavourite=true}

Enter the index of the item to replace: 1


--- CATEGORIES FOR REPLACEMENTS ---

1. Grocery
2. Cooking
3. Electronics
4. Clothing
5. Accessories

Select a category: 
```



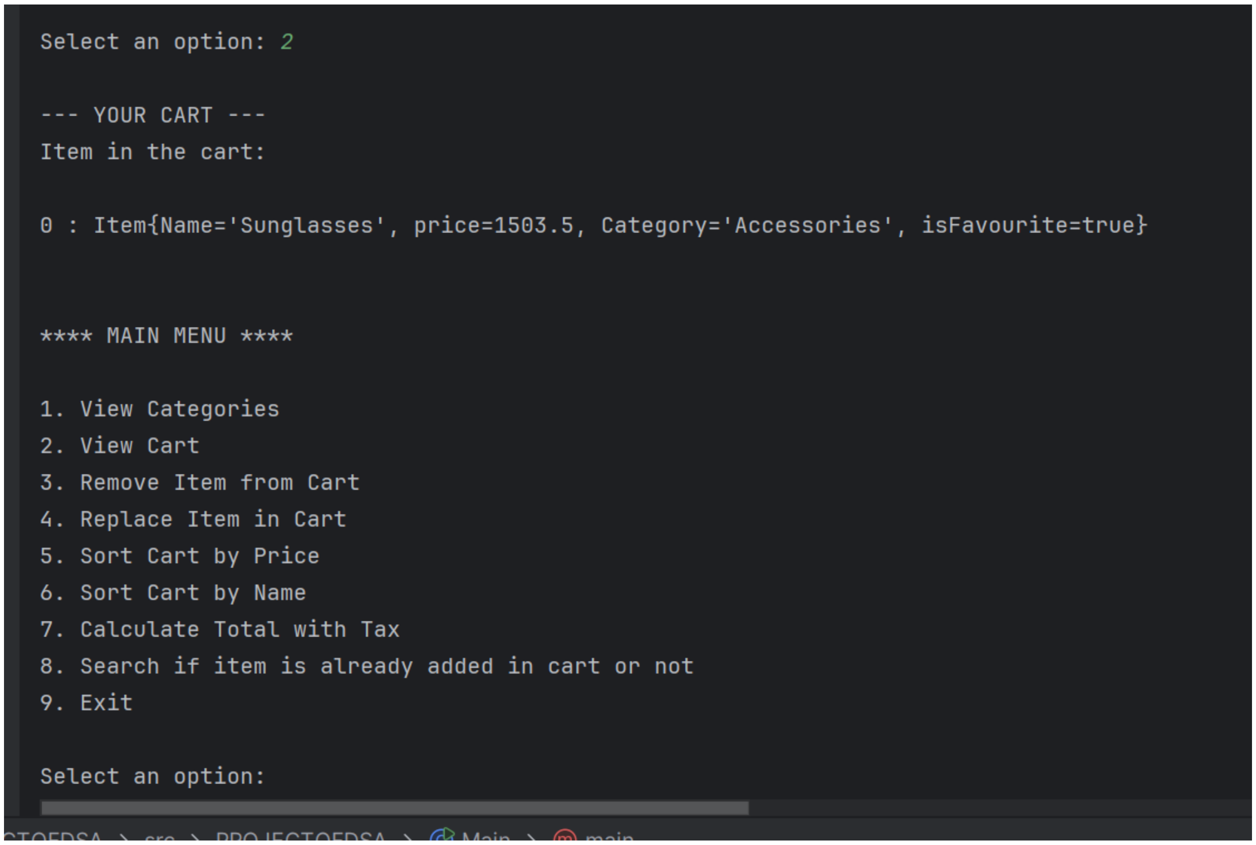
```
Enter the index of the item to replace: 1


--- CATEGORIES FOR REPLACEMENTS ---

1. Grocery
2. Cooking
3. Electronics
4. Clothing
5. Accessories

Select a category: 5


--- ITEMS IN SELECTED CATEGORIES ---
1. Watch - Rs.2502.9 (Available)
2. Sunglasses - Rs.1503.5 (Available)
3. Earrings - Rs.1000.5 (Sold Out)
4. Ring - Rs.1200.9 (Available)
5. Necklace - Rs.2050.1 (Available)

Enter the item number to replace with:
```



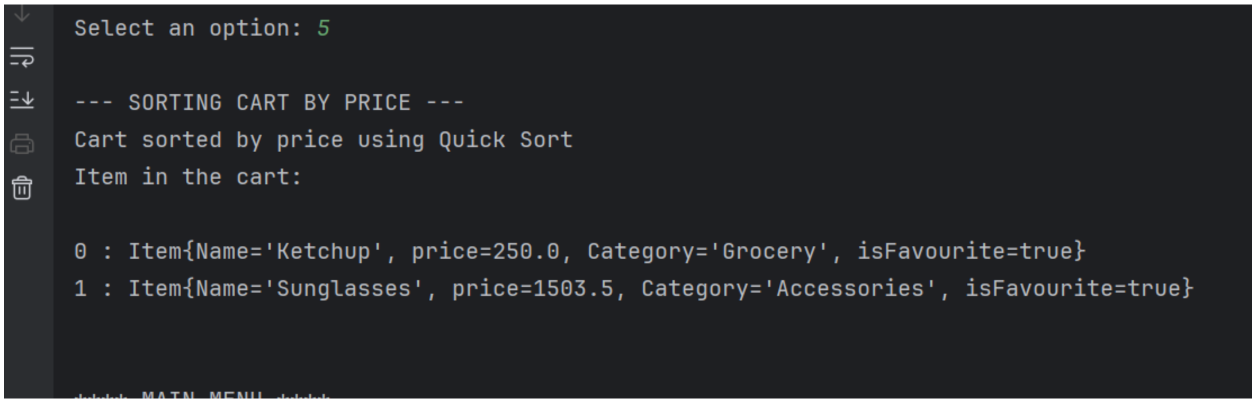
```
4. Ring - Rs.1200.9 (Available)
5. Necklace - Rs.2050.1 (Available)

Enter the item number to replace with: 2
 Index 1 has been replaced  with : Item{Name='Sunglasses', price=1503.5, Category='Accessories', isFavourite=true}

Sunglasses has been replaced in the cart!


**** MAIN MENU ****

1. View Categories
2. View Cart
3. Remove Item from Cart
4. Replace Item in Cart
5. Sort Cart by Price
6. Sort Cart by Name
7. Calculate Total with Tax
8. Search if item is already added in cart or not
9. Exit

Select an option: 
```



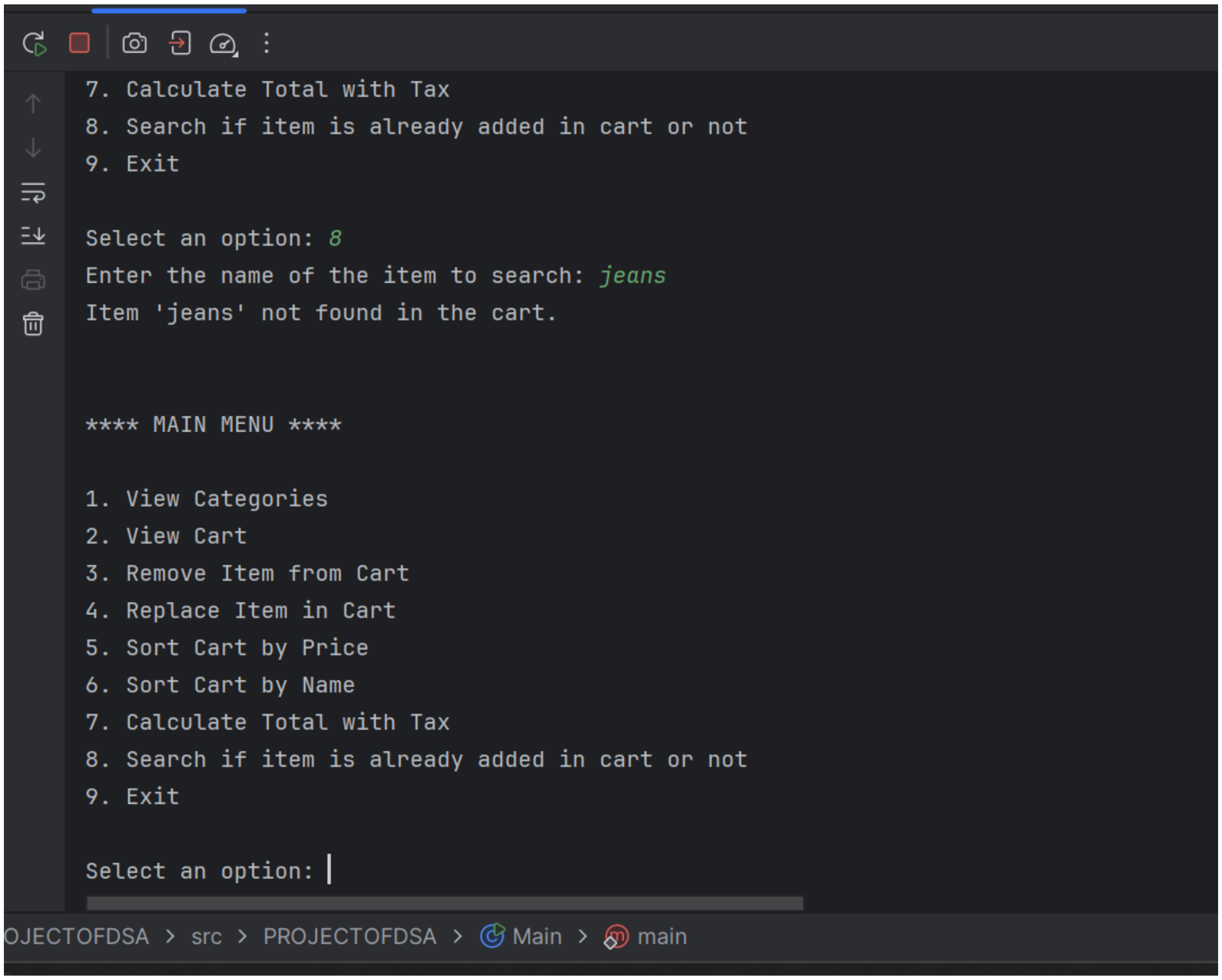
```
--- REMOVE ITEM FROM CART ---
Item in the cart:

0 : Item{Name='Rice', price=120.0, Category='Grocery', isFavourite=true}
1 : Item{Name='Sunglasses', price=1503.5, Category='Accessories', isFavourite=true}
Enter the index of the item to remove: 0
 Item at index 0 has been removed from the cart.


**** MAIN MENU ****

1. View Categories
2. View Cart
3. Remove Item from Cart
4. Replace Item in Cart
5. Sort Cart by Price
6. Sort Cart by Name
7. Calculate Total with Tax
8. Search if item is already added in cart or not
```



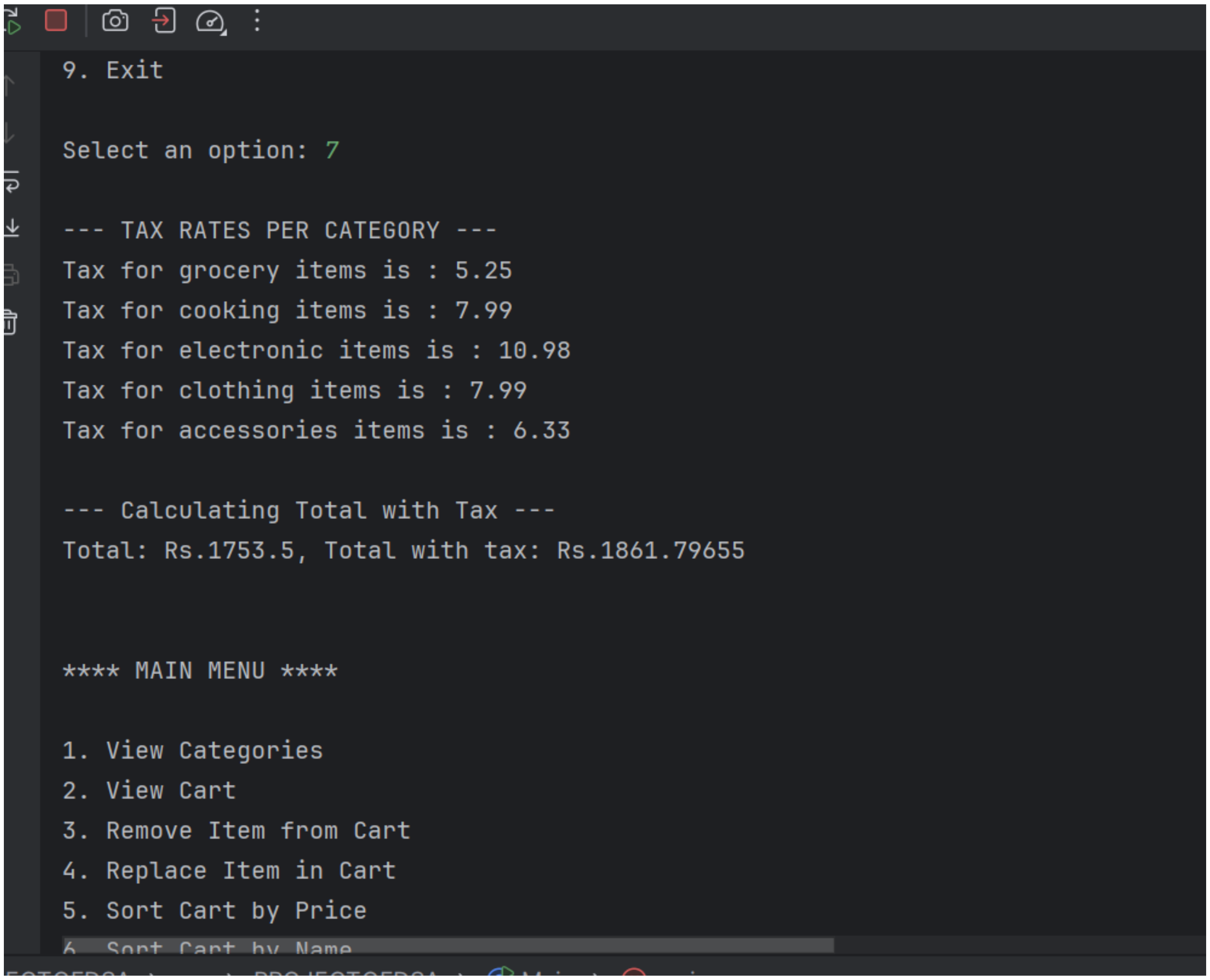
```
Select an option: 2

--- YOUR CART ---
Item in the cart:

0 : Item{Name='Sunglasses', price=1503.5, Category='Accessories', isFavourite=true}


**** MAIN MENU ****

1. View Categories
2. View Cart
3. Remove Item from Cart
4. Replace Item in Cart
5. Sort Cart by Price
6. Sort Cart by Name
7. Calculate Total with Tax
8. Search if item is already added in cart or not
9. Exit

Select an option:
```



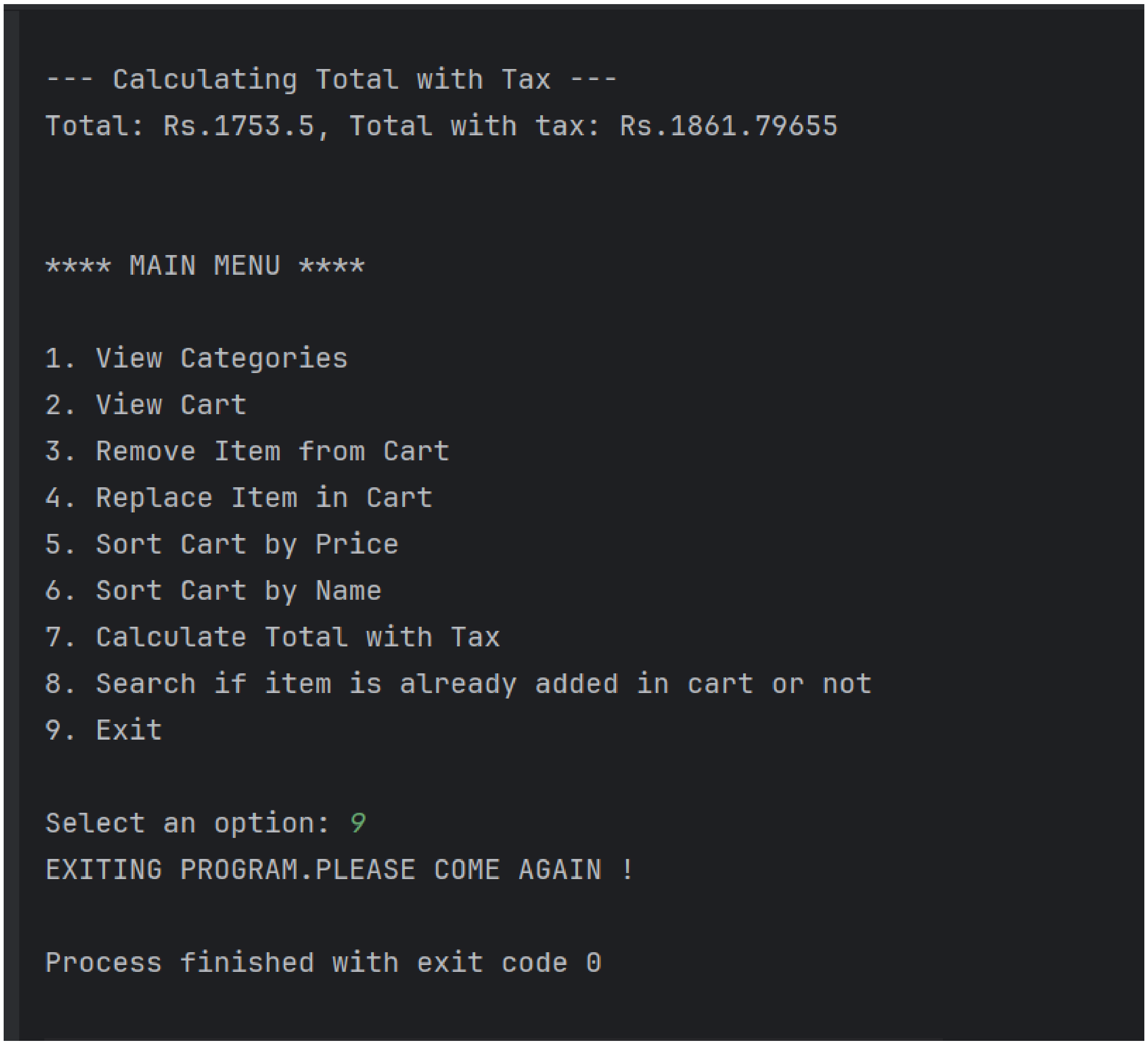
```
Select an option: 5

--- SORTING CART BY PRICE ---
Cart sorted by price using Quick Sort
Item in the cart:

0 : Item{Name='Ketchup', price=250.0, Category='Grocery', isFavourite=true}
1 : Item{Name='Sunglasses', price=1503.5, Category='Accessories', isFavourite=true}
```



```
7. Calculate Total with Tax
8. Search if item is already added in cart or not
9. Exit

Select an option: 8
Enter the name of the item to search: jeans
Item 'jeans' not found in the cart.


**** MAIN MENU ****

1. View Categories
2. View Cart
3. Remove Item from Cart
4. Replace Item in Cart
5. Sort Cart by Price
6. Sort Cart by Name
7. Calculate Total with Tax
8. Search if item is already added in cart or not
9. Exit

Select an option:
```



```
9. Exit

Select an option: 7

--- TAX RATES PER CATEGORY ---
Tax for grocery items is : 5.25
Tax for cooking items is : 7.99
Tax for electronic items is : 10.98
Tax for clothing items is : 7.99
Tax for accessories items is : 6.33

--- Calculating Total with Tax ---
Total: Rs.1753.5, Total with tax: Rs.1861.79655



**** MAIN MENU ****

1. View Categories
2. View Cart
3. Remove Item from Cart
4. Replace Item in Cart
5. Sort Cart by Price
6. Sort Cart by Name
```



```
--- Calculating Total with Tax ---
Total: Rs.1753.5, Total with tax: Rs.1861.79655



**** MAIN MENU ****

1. View Categories
2. View Cart
3. Remove Item from Cart
4. Replace Item in Cart
5. Sort Cart by Price
6. Sort Cart by Name
7. Calculate Total with Tax
8. Search if item is already added in cart or not
9. Exit

Select an option: 9
EXITING PROGRAM.PLEASE COME AGAIN !

Process finished with exit code 0
```

Conclusion

This project demonstrates the effective use of data structures and algorithms in a real-world application, such as a shopping cart system. It highlights the benefits of dynamic arrays, sorting techniques, and stack data structures for efficient data management. The system's scalability and flexibility make it an ideal model for e-commerce platforms, providing a solid foundation for further enhancements.