JAVA ASSIGNMENT : SELF LEARNING COMPONENT
OOADJ Lab – 8
Topic : Java Multi-threading & Serialization (Self Learning)

Name: Siri Gowri H
SRN: PES1UG21CS599
SEC: J

(I) Problem Statement:
Multi-threading
There is a restaurant that needs to assign all of its orders to its chefs. Each dish has a specific amount of time it takes to be prepared, and each chef can only work on one dish at a time. Use multi-threading to get the sequence of completion of the orders, assuming they are being concurrently prepared by all the chefs.

Display messages when an order is assigned to the chef, when the order is in progress, and when the order is ready. In the implementation, it would make things convenient to create a fixed thread pool with the number of threads equal to the number of chefs. You can use the sleep method to simulate the preparation of a dish.

Note that based on the way you assign orders to a chef, and the number of chefs you have in your restaurant, you may have orders completed in a different order. In this example output, I have cyclically assigned orders to each of the 3 total chefs.

PROGRAM:

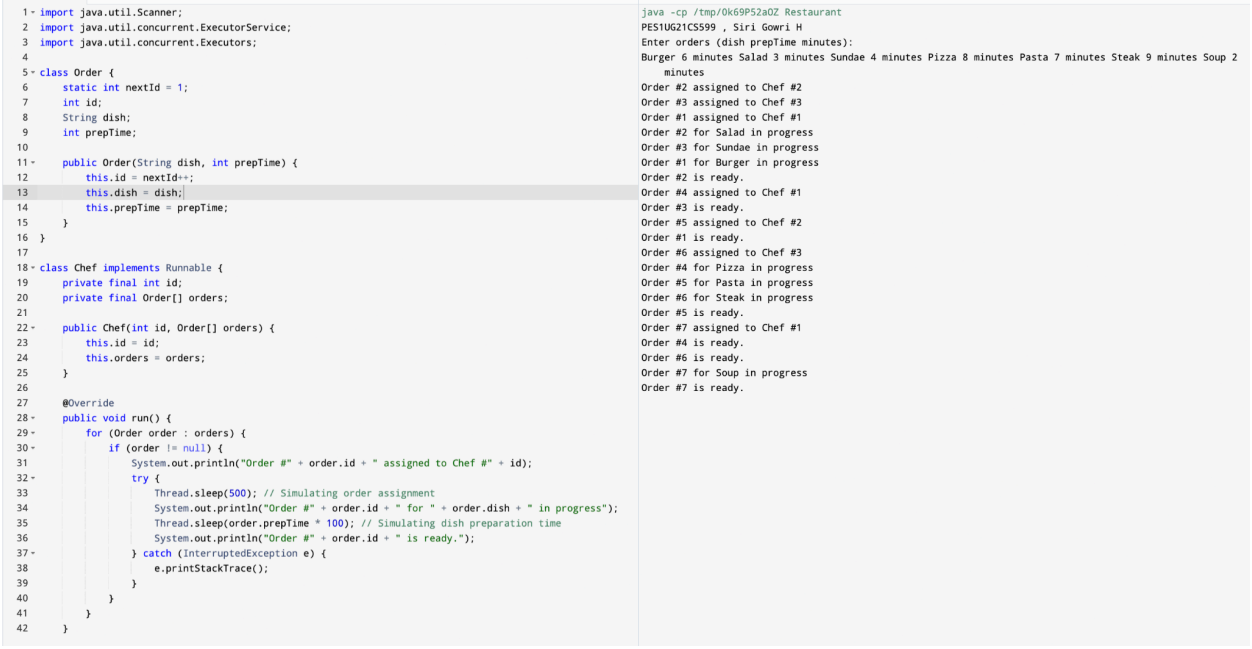
```
import java.util.Scanner;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class Order {
    static int nextId = 1;
    int id;
    String dish;
    int prepTime;

    public Order(String dish, int prepTime) {
        this.id = nextId++;
        this.dish = dish;
        this.prepTime = prepTime;
    }
}
```

```
class Chef implements Runnable {
    private final int id;
    private final Order[] orders;

    public Chef(int id, Order[] orders) {
        this.id = id;
        this.orders = orders;
    }

    @Override
    public void run() {
        for (Order order : orders) {
            if (order != null) {
                System.out.println("Order #" + order.id + " assigned to Chef #" + id);
                try {
                    Thread.sleep(500); // Simulating order assignment
                    System.out.println("Order #" + order.id + " for " + order.dish + " in progress");
                    Thread.sleep(order.prepTime * 100); // Simulating dish preparation time
                    System.out.println("Order #" + order.id + " is ready.");
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
}

class Restaurant {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int numChefs = 3;
        ExecutorService executor = Executors.newFixedThreadPool(numChefs);
        int chefId = 1; // Start the chef IDs from 1
       System.out.println("PES1UG21CS599 , Siri Gowri H ");
        // Taking input for orders
        System.out.println("Enter orders (dish prepTime minutes):");
        while (sc.hasNext()) {
            String dish = sc.next();
            int prepTime = sc.nextInt();
            sc.next(); // Ignore the "minutes" token
            Order order = new Order(dish, prepTime);
            executor.execute(new Chef(chefId++, new Order[]{order})); // Assign a unique ID to each chef
            if (chefId > numChefs) // Reset chef ID counter when it exceeds the number of chefs
                chefId = 1;
        }

        sc.close(); // Close the scanner when done reading input
        executor.shutdown();
```

```
    }
}
```

OUTPUT:

```
PES1UG21CS599 , Siri Gowri H
Enter orders (dish prepTime minutes):
Burger 6 minutes Salad 3 minutes Sundae 4 minutes Pizza 8 minutes Pasta 7 minutes Steak 9 minutes
Soup 2 minutes
Order #2 assigned to Chef #2
Order #3 assigned to Chef #3
Order #1 assigned to Chef #1
Order #2 for Salad in progress
Order #3 for Sundae in progress
Order #1 for Burger in progress
Order #2 is ready.
Order #4 assigned to Chef #1
Order #3 is ready.
Order #5 assigned to Chef #2
Order #1 is ready.
Order #6 assigned to Chef #3
Order #4 for Pizza in progress
Order #5 for Pasta in progress
Order #6 for Steak in progress
Order #5 is ready.
Order #7 assigned to Chef #1
Order #4 is ready.
Order #6 is ready.
Order #7 for Soup in progress
Order #7 is ready.
```



```
import java.util.Scanner;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class Order {
    static int nextId = 1;
    int id;
    String dish;
    int prepTime;

    public Order(String dish, int prepTime) {
        this.id = nextId++;
        this.dish = dish;
        this.prepTime = prepTime;
    }
}

class Chef implements Runnable {
    private final int id;
    private final Order[] orders;

    public Chef(int id, Order[] orders) {
        this.id = id;
        this.orders = orders;
    }

    @Override
    public void run() {
        for (Order order : orders) {
            if (order != null) {
                System.out.println("Order #" + order.id + " assigned to Chef #" + id);
                try {
                    Thread.sleep(500); // Simulating order assignment
                    System.out.println("Order #" + order.id + " for " + order.dish + " in progress");
                    Thread.sleep(order.prepTime * 100); // Simulating dish preparation time
                    System.out.println("Order #" + order.id + " is ready.");
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }
```

```
java -cp /tmp/0k69P52aOZ Restaurant
PES1UG21CS599 , Siri Gowri H
Enter orders (dish prepTime minutes):
Burger 6 minutes Salad 3 minutes Sundae 4 minutes Pizza 8 minutes Pasta 7 minutes Steak 9 minutes Soup 2
    minutes
Order #2 assigned to Chef #2
Order #3 assigned to Chef #3
Order #1 assigned to Chef #1
Order #2 for Salad in progress
Order #3 for Sundae in progress
Order #1 for Burger in progress
Order #2 is ready.
Order #4 assigned to Chef #1
Order #3 is ready.
Order #5 assigned to Chef #2
Order #1 is ready.
Order #6 assigned to Chef #3
Order #4 for Pizza in progress
Order #5 for Pasta in progress
Order #6 for Steak in progress
Order #5 is ready.
Order #7 assigned to Chef #1
Order #4 is ready.
Order #6 is ready.
Order #7 for Soup in progress
Order #7 is ready.
```

ii) Problem Statement: Serialization

Create a command-line application for saving contacts.

A contact must contain a person's name, email address and phone number. The application must have methods for the following:

1. Addanewcontact.
2. Viewallcontacts.
3. Editanexistingcontact. 4. Deleteanexistingcontact.

Make sure the edge cases (deleting or editing an existing contact) are handled properly.

Use serialization for data persistence, so that each time the program is invoked, it reads from a locally-stored address book, and every time the program is exited, the new state of the address book overwrites the previous save.

Look into ObjectInputStream and FileInputStream, and ObjectOutputStream and FileOutputStream for reading from and writing to the local save of the address book. These classes are imported when you import java.io.*.

PROGRAM:

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
public class ContactsApp {
    private static final String SAVE_FILE = "contacts.dat";
    private static List<Contact> contacts = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        loadContacts();
        System.out.println("Loading contacts... Contacts loaded");
        int choice;
        do {
            displayMenu();
            choice = getUserChoice();
            executeChoice(choice);
        } while (choice != 5);
    }
    private static void displayMenu() {
        System.out.println("PES1UG21CS599, SIRI GOWRI H");
        System.out.println("Welcome to Contacts App!");
        System.out.println("1. Add a new contact");
        System.out.println("2. View all contacts");
        System.out.println("3. Edit an existing contact");
        System.out.println("4. Delete an existing contact");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");
    }
    private static int getUserChoice() {
        return scanner.nextInt();
    }
    private static void executeChoice(int choice) {
        switch (choice) {
            case 1:
                addContact();
                break;
            case 2:
                viewAllContacts();
                break;
            case 3:
                editContact();
                break;
            case 4:
                deleteContact();
                break;
            case 5:
                saveContacts();
```

```
                System.out.println("Exiting... Contacts saved");
                break;
            default:
                System.out.println("Invalid choice!");
        }
    }
    private static void loadContacts() {
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(SAVE_FILE))) {
            contacts = (List<Contact>) ois.readObject();
        } catch (FileNotFoundException e) {
            System.out.println("No existing contacts found.");
        } catch (IOException | ClassNotFoundException e) {
            System.out.println("Error loading contacts: " + e.getMessage());
        }
    }
    private static void saveContacts() {
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(SAVE_FILE))) {
            oos.writeObject(contacts);
        } catch (IOException e) {
            System.out.println("Error saving contacts: " + e.getMessage());
        }
    }
   private static void addContact() {
    scanner.nextLine(); // Consume newline
    System.out.print("Enter name: ");
    String name = scanner.nextLine();
    System.out.print("Enter phone number: ");
    String phoneNumber = scanner.nextLine();
    System.out.print("Enter email: ");
    String email = scanner.nextLine();
    contacts.add(new Contact(name, email, phoneNumber));
    System.out.println("Contact added successfully!");
}
    private static void viewAllContacts() {
        if (contacts.isEmpty()) {
            System.out.println("No contacts available.");
        } else {
            System.out.println("Contacts:");
            for (int i = 0; i < contacts.size(); i++) {
                System.out.println((i + 1) + ". " + contacts.get(i));
            }
        }
    }
    private static void editContact() {
        viewAllContacts();
        if (contacts.isEmpty()) {
            return;
        }
        System.out.println("Select a contact to edit:");
```

```
        int index = scanner.nextInt();
        if (index < 1 || index > contacts.size()) {
            System.out.println("Invalid index!");
            return;
        }
        System.out.println("Enter updated details:");
        System.out.print("Enter name: ");
        String name = scanner.next();
        System.out.print("Enter phone number: ");
        String phoneNumber = scanner.next();
        System.out.print("Enter email: ");
        String email = scanner.next();
        Contact contact = contacts.get(index - 1);
        contact.setName(name);
        contact.setPhoneNumber(phoneNumber);
        contact.setEmail(email);
        System.out.println("Contact updated successfully!");
    }
    private static void deleteContact() {
        viewAllContacts();
        if (contacts.isEmpty()) {
            return;
        }
        System.out.println("Select a contact to delete:");
        int index = scanner.nextInt();
        if (index < 1 || index > contacts.size()) {
            System.out.println("Invalid index!");
            return;
        }
        contacts.remove(index - 1);
        System.out.println("Contact deleted successfully!");
    }
    static class Contact implements Serializable {
        private String name;
        private String email;
        private String phoneNumber;
        public Contact(String name, String email, String phoneNumber) {
            this.name = name;
            this.email = email;
            this.phoneNumber = phoneNumber;
        }
        public void setName(String name) {
            this.name = name;
        }
        public void setEmail(String email) {
            this.email = email;
        }
        public void setPhoneNumber(String phoneNumber) {
            this.phoneNumber = phoneNumber;
```

```
        }
        @Override
        public String toString() {
            return "Name: " + name + ", Phone: " + phoneNumber + ", Email: " + email;
        }
    }
}
```

OUTPUT:

```
java -cp /tmp/42DYuEMz0u ContactsApp
Loading contacts... Contacts loaded
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 1
Enter name: John Doe
Enter phone number: 9876403833
Enter email: johndoe@gmail.com
Contact added successfully!
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 1
Enter name: Jane Dsouze
Enter phone number: 93849873249
Enter email: jane@gmail.com
Contact added successfully!
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 2
```

Contacts:
1. Name: Jane Smith, Phone: 9578987630, Email: janesmith@gmail.com
2. Name: John Doe, Phone: 9876403833, Email: johndoe@gmail.com
3. Name: Jane Dsouze, Phone: 93849873249, Email: jane@gmail.com
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 3
Contacts:
1. Name: Jane Smith, Phone: 9578987630, Email: janesmith@gmail.com
2. Name: John Doe, Phone: 9876403833, Email: johndoe@gmail.com
3. Name: Jane Dsouze, Phone: 93849873249, Email: jane@gmail.com
Select a contact to edit:
2
Enter updated details:
Enter name: Gowri
Enter phone number: 9873572892
Enter email: rrgow@gmail.com
Contact updated successfully!
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 4
Contacts:
1. Name: Jane Smith, Phone: 9578987630, Email: janesmith@gmail.com
2. Name: Gowri, Phone: 9873572892, Email: rrgow@gmail.com
3. Name: Jane Dsouze, Phone: 93849873249, Email: jane@gmail.com
Select a contact to delete:
2
Contact deleted successfully!
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact

5. Exit
Enter your choice: 5
Exiting... Contacts saved

```java
import java.util.Scanner;

public class ContactsApp {
    private static final String SAVE_FILE = "contacts.dat";
    private static List<Contact> contacts = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        loadContacts();
        System.out.println("Loading contacts... Contacts loaded");


        int choice;
        do {
            displayMenu();
            choice = getUserChoice();
            executeChoice(choice);
        } while (choice != 5);
    }

    private static void displayMenu()
    {
        System.out.println("PES1UG21CS599, SIRI GOWRI H");
        System.out.println("Welcome to Contacts App!");
        System.out.println("1. Add a new contact");
        System.out.println("2. View all contacts");
        System.out.println("3. Edit an existing contact");
        System.out.println("4. Delete an existing contact");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");
    }

    private static int getUserChoice() {
        return scanner.nextInt();
    }

    private static void executeChoice(int choice) {
        switch (choice) {
            case 1:
                addContact();
                break;
            case 2:
                viewAllContacts();
                break;
            case 3:
                editContact();
                break;
            case 4:
```

```
java -cp /tmp/42DYuEMzOu ContactsApp
Loading contacts... Contacts loaded
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 1
Enter name: John Doe
Enter phone number: 9876403833
Enter email: johndoe@gmail.com
Contact added successfully!
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 1
Enter name: Jane Dsouze
Enter phone number: 93849873249
Enter email: jane@gmail.com
Contact added successfully!
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 2
Contacts:
1. Name: Jane Smith, Phone: 9578987630, Email: janesmith@gmail.com
2. Name: John Doe, Phone: 9876403833, Email: johndoe@gmail.com
3. Name: Jane Dsouze, Phone: 93849873249, Email: jane@gmail.com
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 3
Contacts:
1. Name: Jane Smith, Phone: 9578987630, Email: janesmith@gmail.com
```

```java
import java.util.Scanner;

public class ContactsApp {
    private static final String SAVE_FILE = "contacts.dat";
    private static List<Contact> contacts = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        loadContacts();
        System.out.println("Loading contacts... Contacts loaded");


        int choice;
        do {
            displayMenu();
            choice = getUserChoice();
            executeChoice(choice);
        } while (choice != 5);
    }

    private static void displayMenu()
    {
        System.out.println("PES1UG21CS599, SIRI GOWRI H");
        System.out.println("Welcome to Contacts App!");
        System.out.println("1. Add a new contact");
        System.out.println("2. View all contacts");
        System.out.println("3. Edit an existing contact");
        System.out.println("4. Delete an existing contact");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");
    }

    private static int getUserChoice() {
        return scanner.nextInt();
    }

    private static void executeChoice(int choice) {
        switch (choice) {
            case 1:
                addContact();
                break;
            case 2:
                viewAllContacts();
                break;
            case 3:
                editContact();
                break;
            case 4:
```

```
Contact added successfully!
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 2
Contacts:
1. Name: Jane Smith, Phone: 9578987630, Email: janesmith@gmail.com
2. Name: John Doe, Phone: 9876403833, Email: johndoe@gmail.com
3. Name: Jane Dsouze, Phone: 93849873249, Email: jane@gmail.com
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 3
Contacts:
1. Name: Jane Smith, Phone: 9578987630, Email: janesmith@gmail.com
2. Name: John Doe, Phone: 9876403833, Email: johndoe@gmail.com
3. Name: Jane Dsouze, Phone: 93849873249, Email: jane@gmail.com
Select a contact to edit:
2
Enter updated details:
Enter name: Gowri
Enter phone number: 9873572892
Enter email: rrgow@gmail.com
Contact updated successfully!
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 4
Contacts:
1. Name: Jane Smith, Phone: 9578987630, Email: janesmith@gmail.com
2. Name: Gowri, Phone: 9873572892, Email: rrgow@gmail.com
3. Name: Jane Dsouze, Phone: 93849873249, Email: jane@gmail.com
Select a contact to delete:
2
Contact deleted successfully!
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
```

```
import java.util.Scanner;

public class ContactsApp {
    private static final String SAVE_FILE = "contacts.dat";
    private static List<Contact> contacts = new ArrayList<>();
    private static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        loadContacts();
        System.out.println("Loading contacts... Contacts loaded");


        int choice;
        do {
            displayMenu();
            choice = getUserChoice();
            executeChoice(choice);
        } while (choice != 5);
    }

    private static void displayMenu()
    {
        System.out.println("PES1UG21CS599, SIRI GOWRI H");
        System.out.println("Welcome to Contacts App!");
        System.out.println("1. Add a new contact");
        System.out.println("2. View all contacts");
        System.out.println("3. Edit an existing contact");
        System.out.println("4. Delete an existing contact");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");
    }

    private static int getUserChoice() {
        return scanner.nextInt();
    }

    private static void executeChoice(int choice) {
        switch (choice) {
            case 1:
                addContact();
                break;
            case 2:
                viewAllContacts();
                break;
            case 3:
                editContact();
                break;
            case 4:
```

```
Contacts:
1. Name: Jane Smith, Phone: 9578987630, Email: janesmith@gmail.com
2. Name: John Doe, Phone: 9876403833, Email: johndoe@gmail.com
3. Name: Jane Dsouza, Phone: 93849873249, Email: jane@gmail.com
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 3
Contacts:
1. Name: Jane Smith, Phone: 9578987630, Email: janesmith@gmail.com
2. Name: John Doe, Phone: 9876403833, Email: johndoe@gmail.com
3. Name: Jane Dsouza, Phone: 93849873249, Email: jane@gmail.com
Select a contact to edit:
2
Enter updated details:
Enter name: Gowri
Enter phone number: 9873572892
Enter email: rrgow@gmail.com
Contact updated successfully!
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 4
Contacts:
1. Name: Jane Smith, Phone: 9578987630, Email: janesmith@gmail.com
2. Name: Gowri, Phone: 9873572892, Email: rrgow@gmail.com
3. Name: Jane Dsouza, Phone: 93849873249, Email: jane@gmail.com
Select a contact to delete:
2
Contact deleted successfully!
PES1UG21CS599, SIRI GOWRI H
Welcome to Contacts App!
1. Add a new contact
2. View all contacts
3. Edit an existing contact
4. Delete an existing contact
5. Exit
Enter your choice: 5
Exiting... Contacts saved
```