WELCOME BACK
&
THANK YOU

Advance Java
Crash Course

For TD Bank

MEET YOUR CRASH COURSE TEAM

TANGY F.

CEO

WILLIAM D.

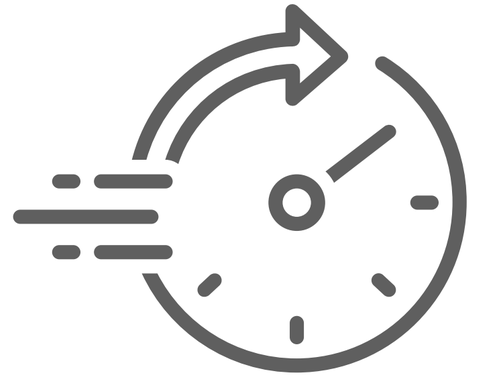
DEVELOPER

TONY J.

T.A * DEVELOPER

Rapid

TRIVIA

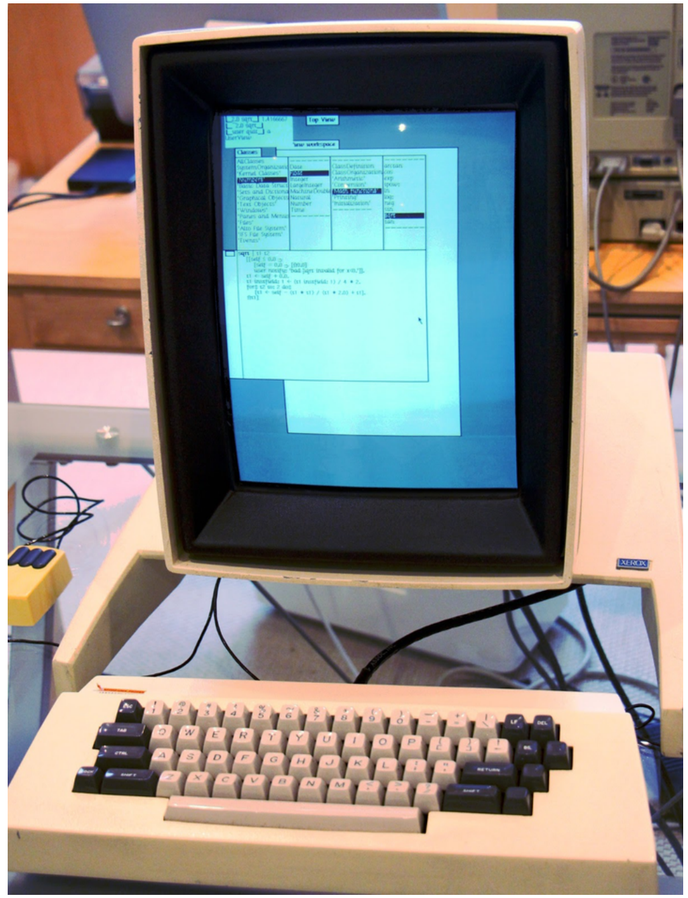


Rapid Trivia

Which was the first computer that incorporated the first graphical user interface and WYSIWYG Text Editing

Rapid

TRIVIA



The Xerox Altos Computer

TRAINING DAY 7

TODAY'S AGENDA

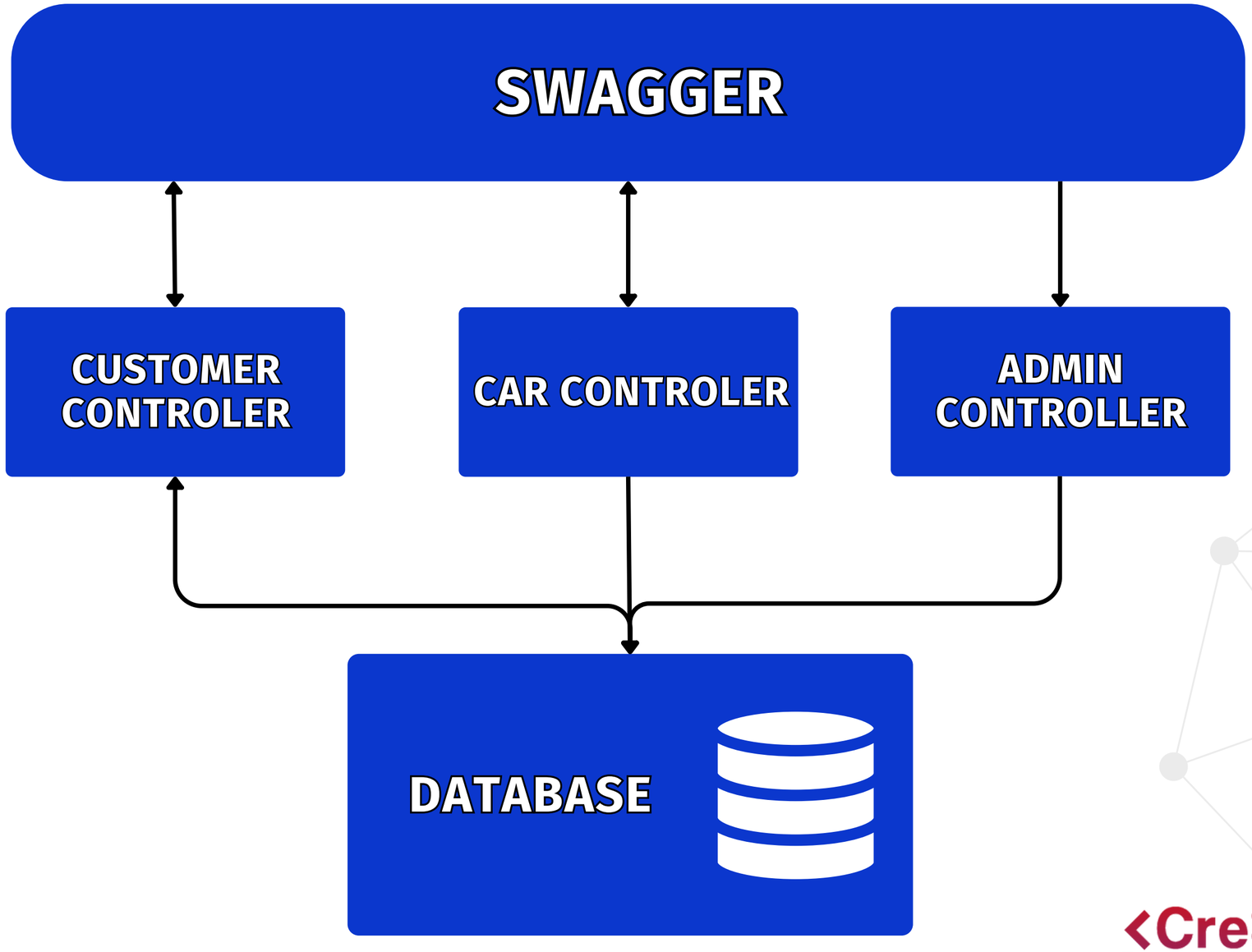
1. Yesterday's Coding Exercise to Go Rapid Review

2. Concurrency
   - Using Join()
   - Using DoubleAdder
   - Using ConcurrentHashMap

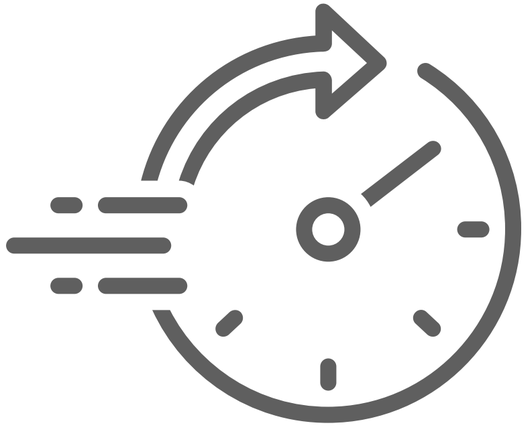
3. Design Patterns. The Template Pattern

BITE SIZE
CAR RENTAL APP

WE ARE BUILDING



Rapid Review
Lesson 1



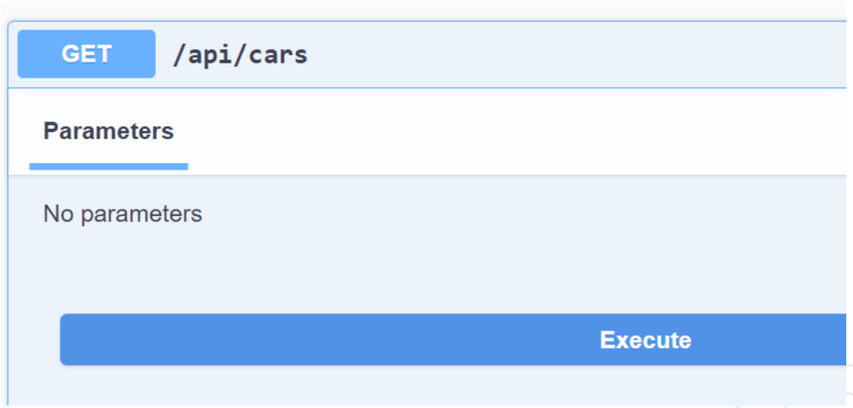
Rapid Review

Adding connection pooling to our cars list API

YESTERDAY'S CODING EXERCISE TO GO

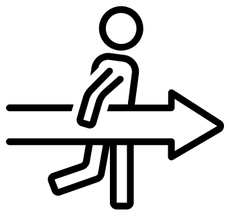
We have seen connection pooling in the DBAccessCP main class. Lets now add it to our autohire app by going to the **DatabaseConfing** class, add a new method **getConnectionPool()** that implements a connection access using connection pool via Hikari. Then ensure that when you run /api/cars, the execution uses **getConnectionPool()** instead of the existing **getConnection()**

```
HikariConfig config = new HikariConfig();
config.setJdbcUrl(jdbcUrl:"jdbc:mysql://localhost:3306/autohire");
config.setUsername(username:"autohire");
config.setPassword(password:"autohire");

// Create a data source using HikariCP
HikariDataSource dataSource = new HikariDataSource(config);

try (Connection connection = dataSource.getConnection()) {
    // Perform database operations using the connection
    String sql = "SELECT * FROM customer";
    try (PreparedStatement preparedStatement = connection.prepareStatement(sql);
         ResultSet resultSet = preparedStatement.executeQuery()) {
        while (resultSet.next()) {
            int customerId = resultSet.getInt("id");
            String name = resultSet.getString("first_name");
            String lastName = resultSet.getString("last_name");
            System.out.println("Customer ID: " + customerId + ", Customer Name: " + name + " " + lastName);
        }
    }
} catch (SQLException e) {
    e.printStackTrace();
} finally {
    // Close the data source to release resources
    dataSource.close();
}
```



YESTERDAY'S CODING TO GO REVIEW

```
public static HikariDataSource getConnectionPool() throws SQLException {
    Properties properties = new Properties();
    try (InputStream inputStream = DatabaseConfig.class.getClassLoader().getResourceAsStream(PROPERTY_FILE_PATH))
        if (inputStream == null) {
            throw new SQLException("Unable to find property file: " + PROPERTY_FILE_PATH);
        }

        properties.load(inputStream);

        String url = properties.getProperty("datasource.url");
        String username = properties.getProperty("datasource.username");
        String password = properties.getProperty("datasource.password");

        HikariConfig config = new HikariConfig();
        config.setJdbcUrl(url);
        config.setUsername(username);
        config.setPassword(password);

        return new HikariDataSource(config);
    } catch (IOException e) {
        throw new SQLException("Failed to load database properties from property file.", e);
    }

}
```



Let's Take a Look.

```
try (// Connection connection = DatabaseConfig.getConnection(); // without connection pooling
    Connection connection = DatabaseConfig.getConnectionPool().getConnection(); // with connection pooling
```

DAY 7
LESSON 2

Concurrency

CONCURRENCY

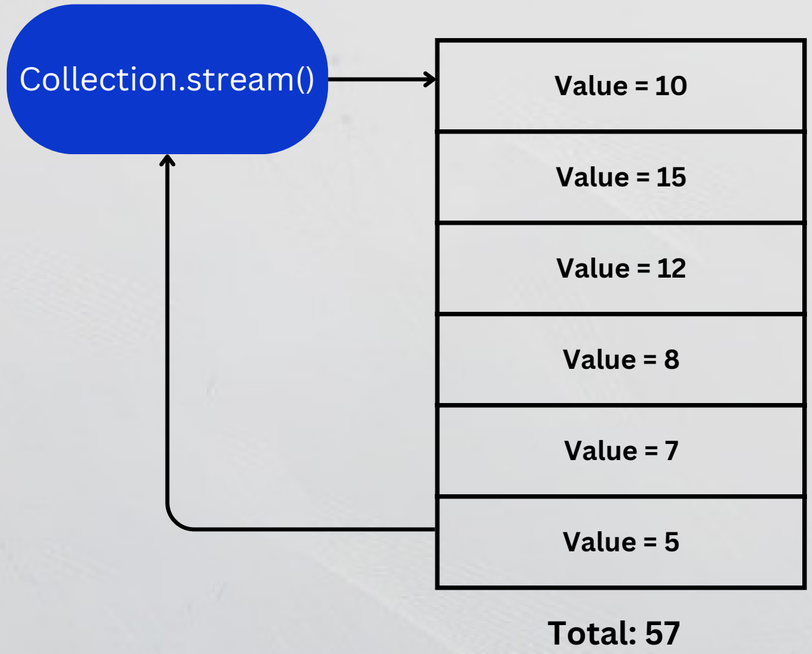
Thread Safety Using join()

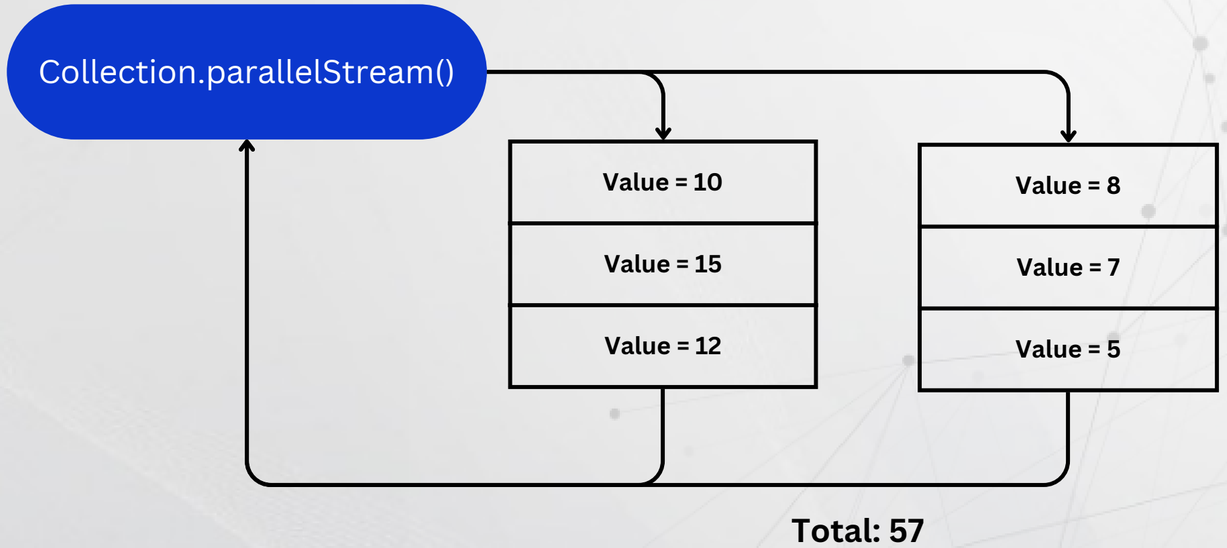
```
Runnable task = () -> {
    for (int i = 0; i < 10; i++) {
        counter.increment();
        System.out.println("Thread " + Thread.currentThread().getId() + ": " + i);
    }
};

Thread thread1 = new Thread(task);
Thread thread2 = new Thread(task);

thread1.start();
thread2.start();

try {
    if (isSafe) {
        thread1.join();
        thread2.join();
    }
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

Lets See it Running

CONCURRENCY

Thread Safety Using DoubleAdder






```
DoubleAdder doubleAdder = new DoubleAdder();

Runnable task = () -> {
    for (int i = 0; i < 1000; i++) {
        doubleAdder.add(0.1); // Add 0.1 to the total
    }
};
```

Lets See it Running

CONCURRENCY

Thread Safety Using ConcurrentHashMap

For thread safety use ConcurrentHashMap

```
Map<Integer, String> concurrentHashMap = new ConcurrentHashMap<>();

Runnable concurrentHashMapTask = () -> {
    for (int i = 0; i < 1000; i++) {
        concurrentHashMap.put(i, "Value " + i);
    }
};
```

Instead of HashMap

```
Map<Integer, String> hashMap = new HashMap<>();

Runnable hashMapTask = () -> {
    for (int i = 0; i < 1000; i++) {
        hashMap.put(i, "Value " + i);
    }
};
```

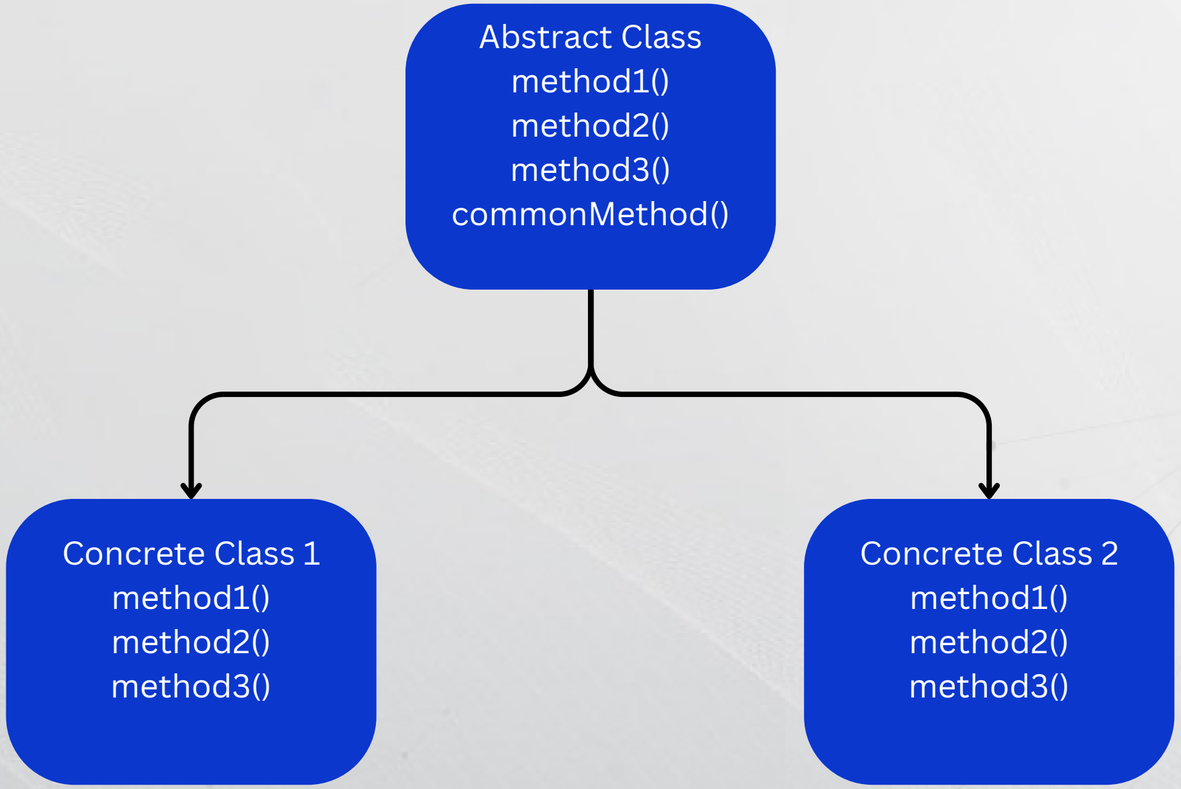
Lets See it Running

DAY 7
LESSON 3

Design Pattern
Template Pattern

DESIGN PATTERN

The Template Pattern



Lets See it Running

CODING TRIVA QUESTION

What does the following piece of code do:

```
List<Client> filteredClients = clients.stream().filter(new java.util.function.Predicate<Client>() {
    @Override
    public boolean test(Client clients) {
        return clients.getAge() >= 30;
    }
}).collect(Collectors.toList());
```

CODING TRIVA ANSWER

The following code take the "clients" list object and filters the list with clients older than 30 years old, into a new object list called filteredClients

```
List<Client> filteredClients = clients.stream().filter(new java.util.function.Predicate<Client>() {
    @Override
    public boolean test(Client clients) {
        return clients.getAge() >= 30;
    }
}).collect(Collectors.toList());
```

CODING EXERCISE TO GO

When using Swagger **customer-controller /api/customer/{customerId}**, the totalPrice is calculated by adding up all the cars the customer have rented. It currently using a traditional **for ()** to loop and sum. Apply parallelStreams to calculate and use **DoubleAdder** for concurrency safety.

Code	Details
200	Response body


```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "js@eall.com",
  "id": 56,
  "totalPrice": 895,
  "carList": [
    {
      "manufacturer": "Tesla",
      "model": "Model S",
```

THANK YOU

<Creative Devs. Software/>

Crash Course

We will see you on Wed.