WELCOME BACK
&
THANK YOU

Advance Java
Crash Course

For TD Bank

MEET YOUR CRASH COURSE TEAM

TANGY F.

CEO

WILLIAM D.

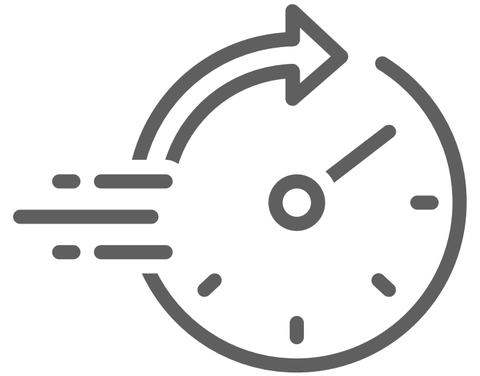
DEVELOPER

TONY J.

T.A * DEVELOPER

Rapid Review

TRIVIA LESSON 1



Rapid Trivia

In which year was the first email sent?

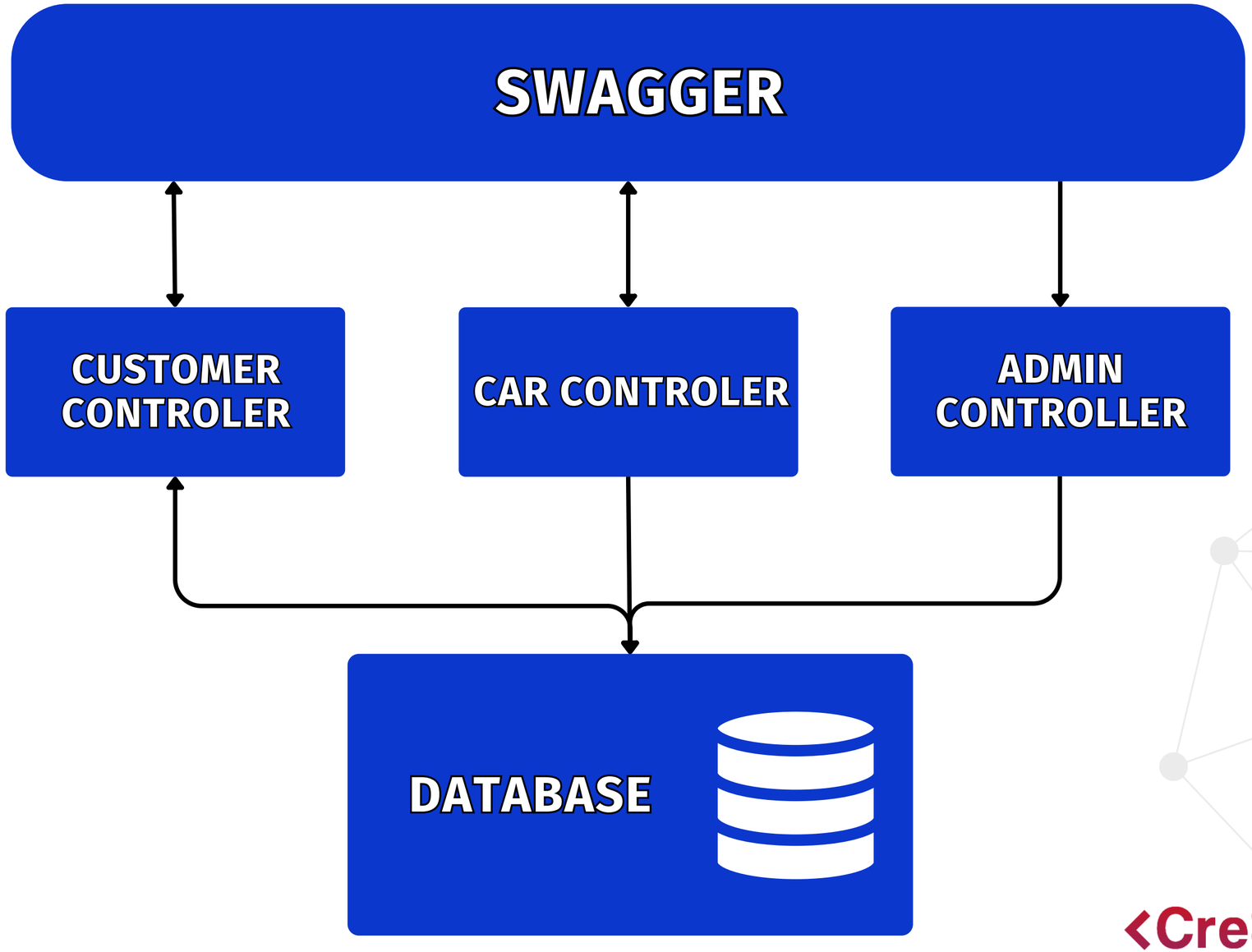
TRAINING DAY 8

TODAY'S AGENDA

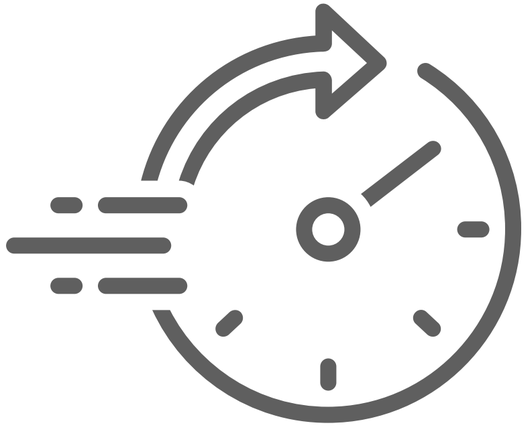
1. Yesterday's Coding Exercise to Go Rapid Review
2. Troubleshooting With Your IDE
3. Java Reflection

BITE SIZE
CAR RENTAL APP

WE ARE BUILDING



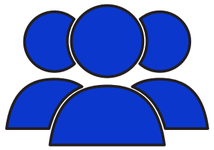
Rapid Review
Lesson 1



Rapid Review

Adding safe concurrency to totalPrice calculation

YERSTERDAYS 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",
```

YESTERDAY'S CODING TO GO REVIEW

```
//double totalPrice = 0.0;
//List<CarResponseDto> cars = (List<CarResponseDto>) customer.getCarList();

/* for (CarResponseDto car : cars) {
    totalPrice += car.getPrice();
}

customer.setTotalExpense(totalPrice); */
```

Code	Details
200	Response body


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


```
DoubleAdder totalPrice = new DoubleAdder();
customer.getCarList().parallelStream()
        .forEach(car -> totalPrice.add(car.getPrice()));


customer.setTotalExpense(totalPrice.doubleValue());
```

DAY 8
LESSON 2

Troubleshooting Code With Your IDE

TROUBLESHOOTING WITH YOUR IDE

Breakpoints

```
28    public ResponseEntity<List<CarResponseDto>> getAllCars() {
29        List<CarResponseDto> cars = carService.getAllCars(); carService = CarService@104
30        return new ResponseEntity<>(cars, HttpStatus.OK);
```

Conditional Breakpoints

```
29        List<CarResponseDto> cars = carService.getAllCars();
```

Expression: `carService.getAllCars() == null;`

Inspecting Variables and changing variables values

```
WATCH
> carId: Long@155 "3"
```

```
WATCH
> carId = (long) 2: Long@170 "2"
```

Call Stack

```
Thread [http-nio-...  PAUSED ON BREAKPOINT
    CarService.getCarById(Long)        CarService.java
    CarController.getCarById(Long)     CarController...
```

Object List

```
VARIABLES
Local
> carId: Long@43 "3"
> selectQuery: "SELECT * FROM car WHERE id = ?
> connection: ConnectionImpl@81
  preparedStatement: ClientPreparedStatement
> this: CarService@41
```

Breakpoint List

```
BREAKPOINTS
  Uncaught Exceptions
  Caught Exceptions
  CarController.java   src\main\java\com\carrental\a...  23
  CarController.java   src\main\java\com\carrental\a...  30
  CarController.java   src\main\java\com\carrental\a...  36
  CarService.java      src\main\java\com\carrental\aut...  115
```

Work together

Let's Take a Look.

DAY 8
LESSON 2

Java Reflection

JAVA REFLECTION

Instantiating when class name is not known at development time.

```
String className = "com.mains.Reflection.TheClass";
Class<?> clazz = Class.forName(className);
ClassInterface instance = (ClassInterface) clazz.getDeclaredConstructor().newInstance();
System.out.println("Id = " + instance.getId() + " Name = " + instance.getName() + " Number = " + instance.getNumber());
```

Print Modifiers

```
int modifiers = clazz.getModifiers();
System.out.println("Modifiers: " + Modifier.toString(modifiers));
```

Print Declared Fields

```
Field[] fields = clazz.getDeclaredFields();
for (Field field : fields) {
    System.out.println("  " + field.getType().getSimpleName() + " " + field.getName());
}
```

Print Declared Methods

```
Method[] methods = clazz.getDeclaredMethods();
for (Method method : methods) {
    System.out.print("  " + method.getReturnType().getSimpleName() + " " + method.getName() + "(");
    Class<?>[] parameterTypes = method.getParameterTypes();
    for (int i = 0; i < parameterTypes.length; i++) {
        System.out.print(parameterTypes[i].getSimpleName());
        if (i < parameterTypes.length - 1) {
            System.out.print(", ");
        }
    }
    System.out.println(")");
}
```

Lets See it Running

CODING TRIVA QUESTION

In our autohire application, Java reflection is used twice. Find out both locations class and method name.

Hint: we have discussed one of them in a previous session.

CODING TRIVA ANSWER

CarRequestValidator

```
public static boolean isValidCarRequestDto(CarRequestDto dto) throws IllegalAccessException {
    Field[] fields = dto.getClass().getDeclaredFields();
```

CustomerRequestValidator

```
public static boolean isValidCustomerRequestDto(BookVehicleDto dto) throws IllegalAccessException {
    Field[] fields = dto.getClass().getDeclaredFields();
```

Lets See the Code

CODING EXERCISE TO GO

In our **com.mains.Reflection.ReflectionMain** class we instantiate a class defined in **application.properties class.className**

Create another class like **TheClass** but should return different values in their methods. Change **application.properties** to reflect the new class and execute **com.mains.Reflection.ReflectionMain**. Confirm the new values for **Id**, **Name** and **Number** should print in the terminal output.

application.properties

```
class.className=com.mains.Reflection.TheClass
```

THANK YOU

Crash Course

We will see you tomorrow