WELCOME
&
THANK YOU

<Creative Devs.
Software/>

# Advance Java Crash Course

For TD Bank

# MEET A FEW OF OUR TEAM MEMBERS

TANGY F.
CEO

LINA G
PROJECT MANAGER

CHEDDAE G.
TECH ASSISTANT

WILLIAM D.
DEVELOPER

TONY
DEVELOPER

- *Founded in South Florida in 2018*
- *Evolved from Software Development Expanded into a Technical Training firm Providing tailored technical training solutions for businesses*

# THE CRASH COURSE

Crash Course is our
bite size lighting training

The Duration is 1 hour a day
for 10 days

One Topic~
In this case it's Advance Java

<Creative Devs. Software/>

# WE LISTENED TO YOU & YOUR COLLEAGUES

1. We interviewed practitioners and your leads
2. We reviewed anonymous data
3. Came up with a tailored curriculum based on real-time data

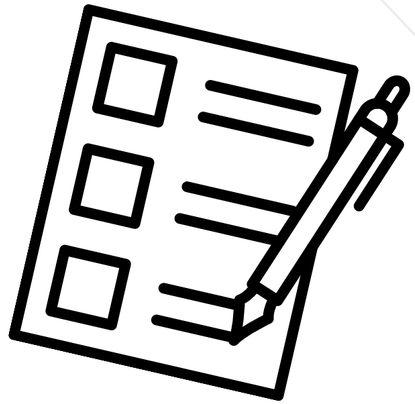
# THE SPACE

A safe space where our goal is to help you

We are here to help you grow. If there is something you don't understand this is the place to ask.

This is a 'No Judgment Zone'

TIME SET ASIDE



# Pre Assessment

# MEET YOUR CRASH COURSE TEAM

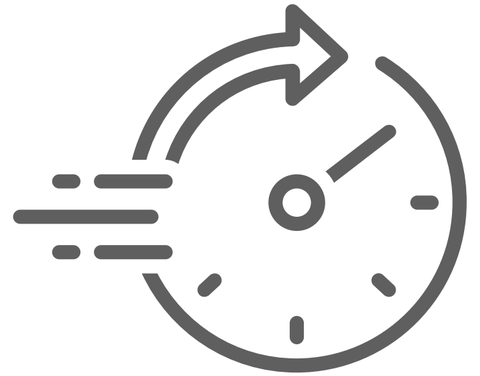
**TANGY F.**
**CEO**

**WILLIAM D.**
**DEVELOPER**

**TONY J.**
**TA & DEVELOPER**

Rapid Review

TRIVIA



# Rapid Trivia

**What was the first language that popularized object oriented programming**

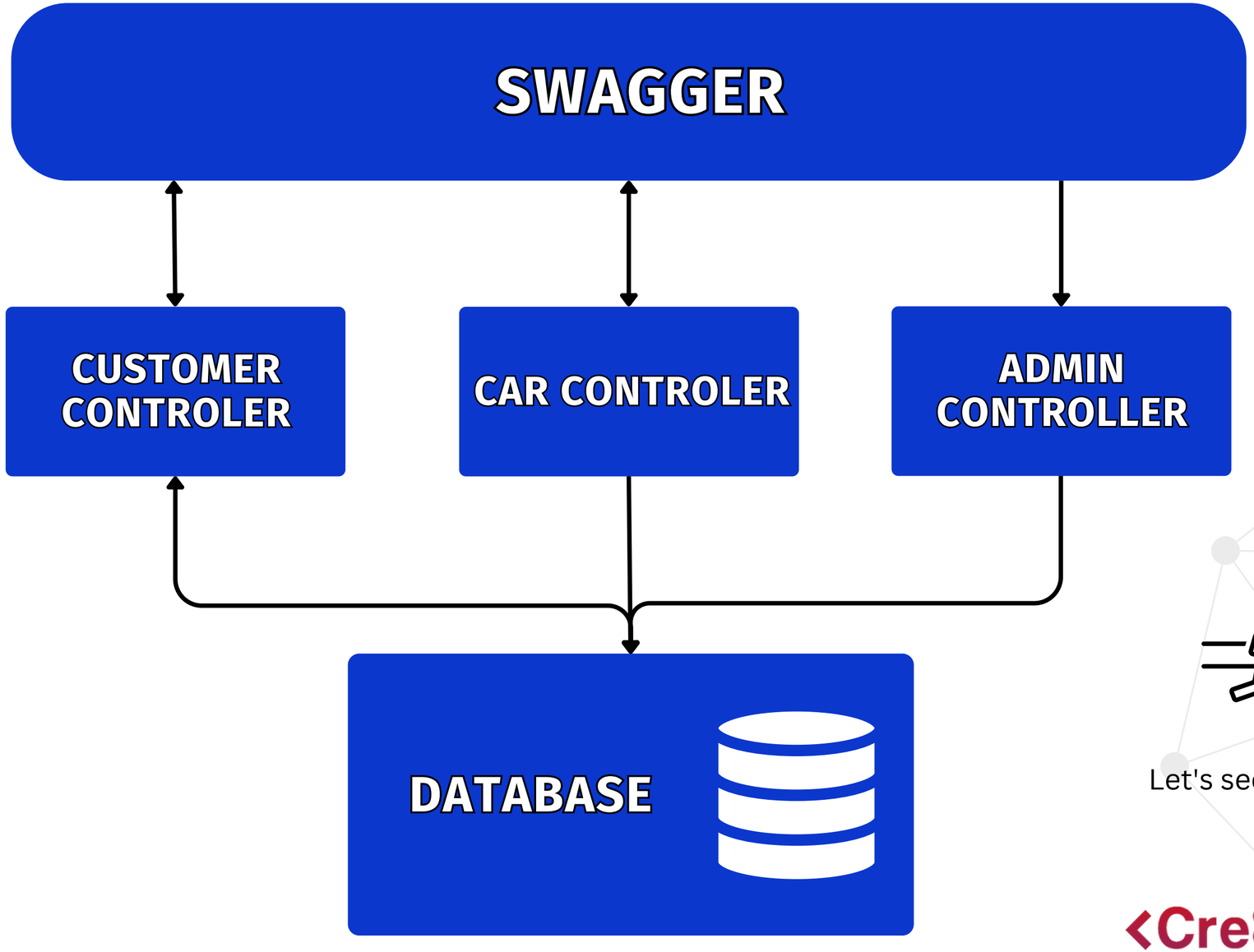
# TRAINING DAY 1

## TODAY'S AGENDA

1. What we are building
2. Java Class Loader
3. Java Just in Time Compiler
4. Maven vs. Gradle
5. Exeption Handling

# BITE SIZE
## CAR RENTAL APP

# WE ARE BUILDING




Let's see it running.

DAY 1
LESSON 2 - 3

# Java Class Loader and JIT Compiler

# JAVA CLASS LOADER

ClassLoader Delegation Model

## BOOTSTRAP CLASS LOADER

Loads Core Java Classes

Let's see the code

## EXTENSIONS CLASS LOADER

Loads from jre/lib/ext

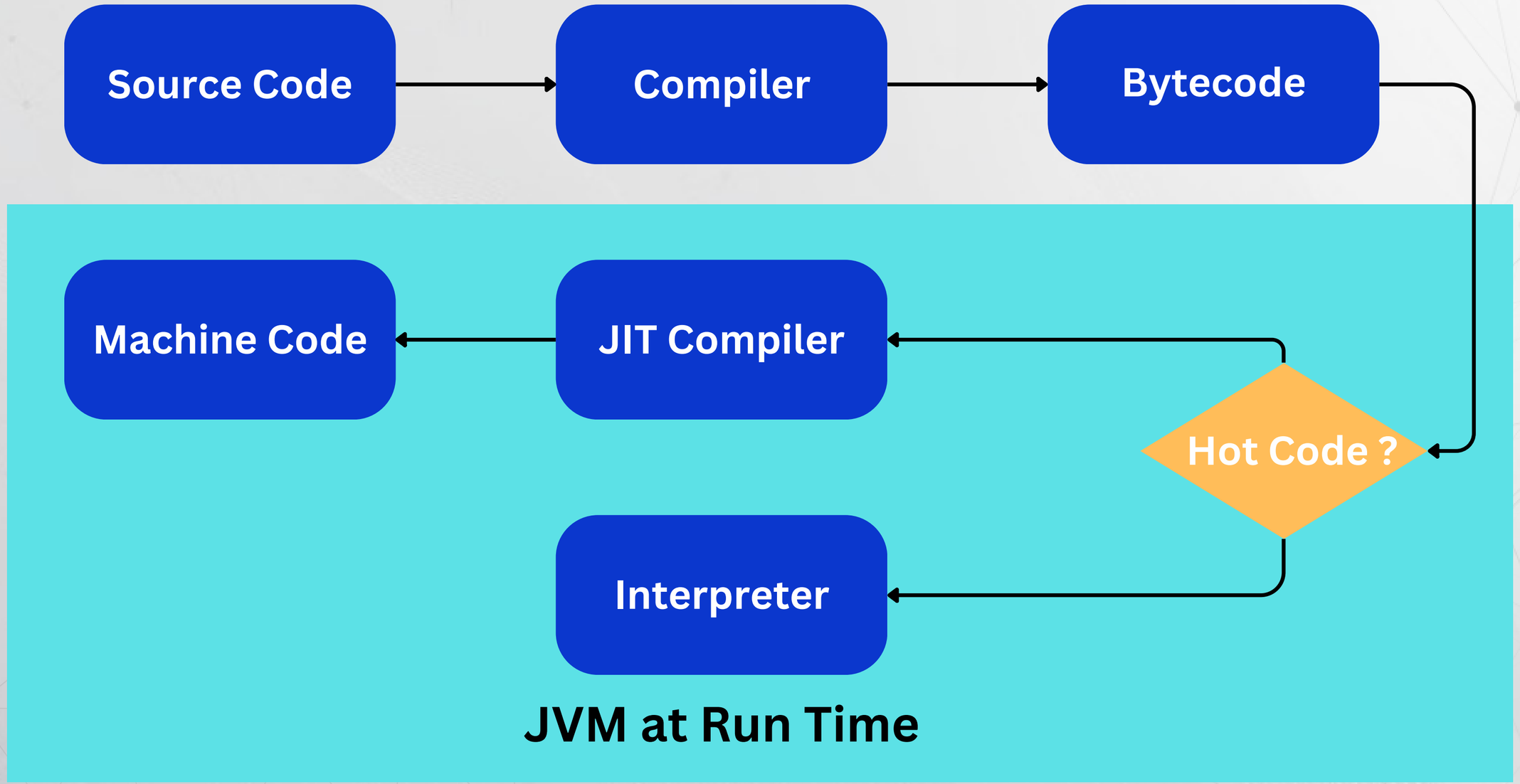
Let's see the code

## APPLICATION CLASS LOADER

Loads From Classpath

Let's see the code

# JAVA JIT COMPILER




Let's see the code

# DAY 1
# LESSON 4

# Gradle

# MAVEN VS. GRADLE

| MAVEN | GRADLE |
| --- | --- |
| Uses XML | Uses DSL |
| Configuration file called "pom.xml" | Configuration file called "build.gradle" |
| Scripts are not as short or clean | Script are shorter and clean |
| Convention over Configuration | Convention over Configuration |
| IDE and Command Line | IDE and Command Line |

# MAVEN VS. GRADLE

Maven POM.xml file dependency for MySql connector

```xml
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
</dependency>
```

Gradle build.gradle file dependency for MySql connector

```
runtimeOnly 'com.mysql:mysql-connector-j'
```

# GRADLE FILE

```
plugins { // defines the plugins applied to the project.
    id 'java'
    id 'org.springframework.boot' version '3.1.1'
    id 'io.spring.dependency-management' version '1.1.0'
}

group = 'com.carrental.autohire' // Define a logical group or namespace for your project's artifacts.
version = '0.0.1-SNAPSHOT' // version for this project

java { // Java version to be used for source validation and compilation
    sourceCompatibility = '17'
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor // Extends or inherits the dependencies of another configuration
    }
}

repositories { // From where to obtain dependencies
    mavenCentral()
}
```

# GRADLE FILE CONTINUATION

```
dependencies { // declare the external dependencies, libraries, frameworks and plugins that your project depends on.
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.modelmapper:modelmapper:3.1.1'
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.1.0'
    implementation 'org.xhtmlrenderer:flying-saucer-pdf:9.1.22'
    implementation 'com.sun.mail:jakarta.mail:2.0.1'
    implementation 'javax.activation:activation:1.1.1'
//  implementation 'org.springframework.boot:spring-boot-starter-data-jpa'

}

tasks.named('test') {
    useJUnitPlatform()  // use the junit platform for running tests
}
```

Let's see the code

# GRADLE COMMANDS

**`gradle build`**

- Compiles and assembles your project.

**`gradle clean`**

- Deletes the build directory, removing all build artifacts and intermediate files.

**`gradle test`**

- Executes tests for your project.

**`gradlew [task]`**

- Runs a specific gradle task

**`gradle dependencies`**

- Displays the list of project dependencies and their versions.

DAY 1
LESSON 5

# Exception Handling

# EXCEPTION HANDLING

```java
package com.carrental.autohire.exceptions;

public class InvalidArgumentException extends RuntimeException {
    private String message;
    public InvalidArgumentException(String message) {
        super(message);
        this.message = message;
        System.out.println(this.message);
    }
}
```

Let's see it running.

```java
@PostMapping(value = "bookvehicle")
public ResponseEntity<CustomerResponseDto> bookcar(@RequestBody BookVehicleDto bookVehicleDto) throws IllegalAccessException {
    if (!CustomerRequestValidator.isValidCustomerRequestDto(bookVehicleDto)) {
        throw new InvalidArgumentException(message:"Please check , email , firstname , lastname and email should be in proper format");
    }
    CustomerResponseDto createdCustomer = customerService.bookCar(bookVehicleDto);
    return new ResponseEntity<>(createdCustomer, HttpStatus.CREATED);
}
```

# CODING TRIVA QUESTION

There is a problem in the gradle dependency list bellow. An important component of our app is present during development but not after deployment. Can you spot which one and why.

```
dependencies { // declare the external dependencies, libraries, frameworks and plugins that your project depends on.
    compileOnly 'org.projectlombok:lombok'
    implementation 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    compileOnly 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.modelmapper:modelmapper:3.1.1'
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.1.0'
    implementation 'org.xhtmlrenderer:flying-saucer-pdf:9.1.22'
    implementation 'com.sun.mail:jakarta.mail:2.0.1'
    implementation 'javax.activation:activation:1.1.1'
    implementation 'com.netflix.hystrix:hystrix-core:1.5.18'
}
```

# CODING TRIVIA ANSWER

Answer;
The "org.springframework.boot:spring-boot-starter-web" is the component missing after deployment because it should have used "implementation" instead of "compileOnly"

```
dependencies { // declare the external dependencies, libraries, frameworks and plugins that your project depends on.
    compileOnly 'org.projectlombok:lombok'
    implementation 'com.mysql:mysql-connector-j'
    annotationProcessor 'org.projectlombok:lombok'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.modelmapper:modelmapper:3.1.1'
    implementation 'org.springdoc:springdoc-openapi-starter-webmvc-ui:2.1.0'
    implementation 'org.xhtmlrenderer:flying-saucer-pdf:9.1.22'
    implementation 'com.sun.mail:jakarta.mail:2.0.1'
    implementation 'javax.activation:activation:1.1.1'
    implementation 'com.netflix.hystrix:hystrix-core:1.5.18'
}
```

# CODING EXERCISE TO GO

In our **admin-controler /admin/addcar,** when we add zero to "year", it is not returning "Invalid car details" as expected. Find the right location in the code and make the necessary correction.

Hint: Go to the **AdminController.java** (bellow), place a brakepoint in line 25 and step inside **isValidCarRequestDto** and study the execution to find out.

**admin-controller**

**POST** `/admin/addcar`

**Parameters**

No parameters

**Request body** required

```
{
  "manufacturer": "Mazda",
  "model": "Model 6",
  "year": 0,
  "color": "Red",
  "price": 223,
  "isBooked": false
}
```

```
23      @PostMapping(value = "addcar")
24      public ResponseEntity<? extends AbstractCarDto> addCar(@RequestBody CarRequestDto carRequestDto) throws IllegalAccessException {
25          if (!CarRequestValidator.isValidCarRequestDto(carRequestDto)) {
26              throw new InvalidArgumentException(message:"Invalid car details");
27          }
28          CarResponseDto createdCar = carService.addCar(carRequestDto);
29          return new ResponseEntity<>(createdCar, HttpStatus.CREATED);
30      }
31  }
```

# THANK YOU


## Crash Course

## We will see you tomorrow