WELCOME BACK
&
THANK YOU

Advance Java
Crash Course

For TD Bank

MEET YOUR CRASH COURSE TEAM

TANGY F.

CEO

WILLIAM D.

DEVELOPER

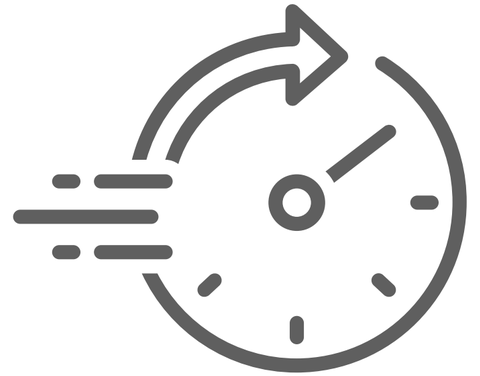
TONY J.

T.A * DEVELOPER

Creative Devs. Software
MOBILE APPS • WEBSITES • AR • VIDEO GAMES • SECURITY

Rapid Review

TRIVIA



Rapid Trivia

Which company developed the java language

Creative Devs.
Software/>
MOBILE APPS•WEBSITES•AR•VIDEO GAMES•SECURITY

TRAINING DAY 4

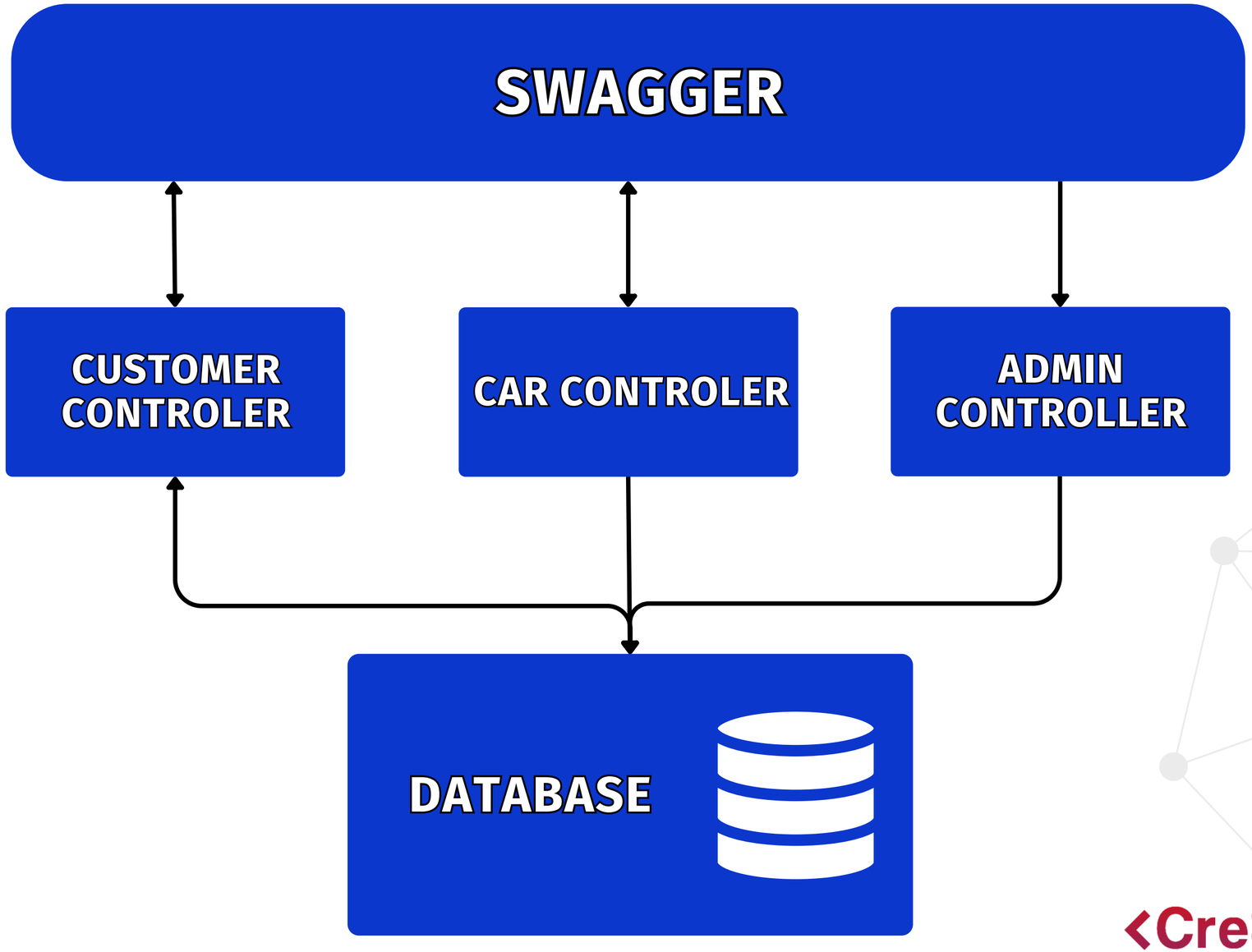
TODAY'S AGENDA

1. Yesterday's Coding Exercise to Go Rapid Review
2. Design Patterns. The Bridge Pattern
3. Wildcards
4. Iterator Design Pattern
5. Garbage Collections

Creative Devs. Software

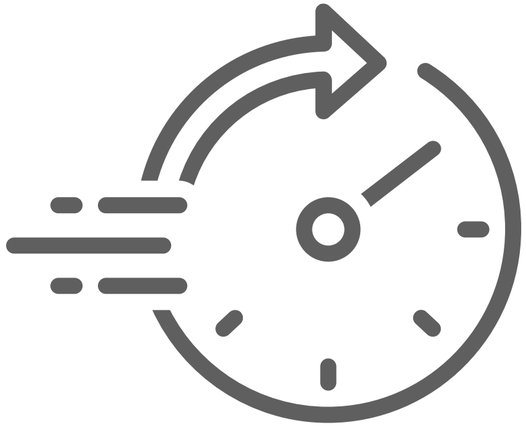
BITE SIZE
CAR RENTAL APP

WE ARE BUILDING



Creative Devs. Software

Rapid Review
Lesson 1



Rapid Review

Add circuit breaker to unregister car

Creative Devs
Software
MOBILE APPS•WEBSITES•AR•VIDEO GAMES•SECURITY

YERSTERDAYS CODING EXERCISE TO GO

Create an UnregisterCircuitBreaker class that extends HystrixCommand<ResponseEntity<String>>. Use this class to add a circuit breaker to **unregister** API. You can use SampleBreaker bellow left as guide.

Use breakpoints to test and you should receive a return like the one bellow

```java
package com.mains.CircuitBraker;
import com.netflix.hystrix.HystrixCommand;
import com.netflix.hystrix.HystrixCommandGroupKey;

class SampleBreaker extends HystrixCommand<String> {
    SampleBreaker() {
        super(HystrixCommandGroupKey.Factory.asKey(name:"SampleGroup"));
    }

    @Override
    protected String run() {
        return "Service response";
    }

    @Override
    protected String getFallback() {
        return "Fallback response";
    }
}
```

500
Undocumented

Error: response status is 500

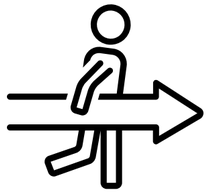
Response body

```
Fallback Response
```

YESTERDAY'S CODING TO GO REVIEW

src > main > java > com > carrental > autohire > service > UnregisterCircuitBreaker.java > UnregisterCircuitBreaker > run()

```java
package com.carrental.autohire.service;
import com.netflix.hystrix.HystrixCommand;
import com.netflix.hystrix.HystrixCommandGroupKey;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;

public class UnregisterCircuitBreaker extends HystrixCommand<ResponseEntity<String>> {
    private CarService carService;
    private Long carId;
    public UnregisterCircuitBreaker(CarService carService, Long carId) {
        super(HystrixCommandGroupKey.Factory.asKey(name:"CarGroup"));
        this.carService = carService;
        this.carId = carId;
    }

    @Override
    protected ResponseEntity<String> run() {
        // execute potentially risky situation
        return new ResponseEntity<>(carService.unregisterCarById(carId), HttpStatus.OK);
    }

    @Override
    protected ResponseEntity<String> getFallback() {
        // Fallback response in case of failure
        return new ResponseEntity<>(body:"Fallback Response", HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```



Let's see it running.

```java
@PostMapping("unregister/{carId}")
public ResponseEntity<String> unregisterCar(@PathVariable Long carId) {
    return new UnregisterCircuitBreaker(carService, carId).execute();
   //return new ResponseEntity<>(carService.unregisterCarById(carId), HttpStatus.OK);
}
```

Creative Software
MOBILE APPS • WEBSITES • AR • VIDEO GAMES • SECURITY

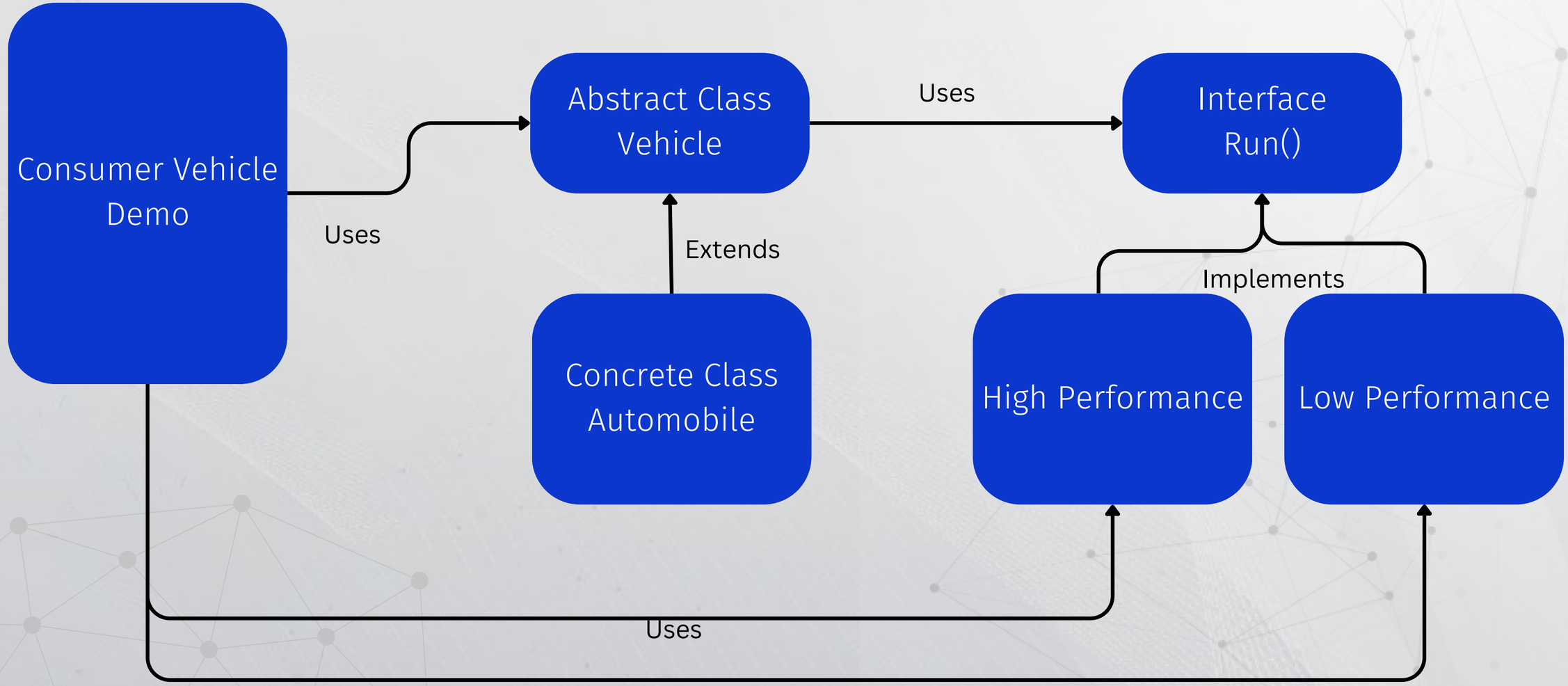
DAY 4
LESSON 2

Design Patterns
The Bridge Pattern

Creative Devs
Software
MOBILE APPS•WEBSITES•AR•VIDEO GAMES•SECURITY

UNDERSTANDING DESIGN PATTERNS

Bridge Pattern



UNDERSTANDING DESIGN PATTERNS

Bridge Pattern High Level Abstraction

```java
package com.carrental.BridgePattern;
public abstract class Vehicle {
    protected RunAPI runAPI;

    protected Vehicle(RunAPI runAPI){
        this.runAPI = runAPI;
    }
    public abstract void run();
}
```

```java
package com.carrental.BridgePattern;
public class Car extends Vehicle {

    public Car(RunAPI runAPI) {
        super(runAPI);
    }

    public void run() {  // implement high level logic here
        runAPI.runVehicle(vehicleType:"Car");
    }
}
```

UNDERSTANDING DESIGN PATTERNS

Bridge Pattern Low Level Implementation

```java
package com.carrental.BridgePattern;
public interface RunAPI {
    public void runVehicle(String vehicleType);
}
```

```java
package com.carrental.BridgePattern;
public class HighPerformance implements RunAPI {
    @Override
    public void runVehicle(String vehicleType) { // Implement specific logic here
        System.out.println(vehicleType + " is High Performance");
    }
}
```

```java
package com.carrental.BridgePattern;
public class LowPerformance implements RunAPI {
    @Override
    public void runVehicle(String vehicleType) { // Implement specific logic here
        System.out.println(vehicleType + " is Low Performance");
    }
}
```

UNDERSTANDING DESIGN PATTERNS

Bridge Pattern Consumer Class.

```java
package com.carrental.BridgePattern;

import java.util.ArrayList;

public class VehicleDemo {
    Run | Debug
    public static void main(String[] args) {

        ArrayList <Vehicle> vehicleList = new ArrayList<>();
        vehicleList.add(new Car(new HighPerformance()));
        vehicleList.add(new Car(new LowPerformance()));

        for (Vehicle vList : vehicleList) {
            vList.run();
        }
    }
}
```

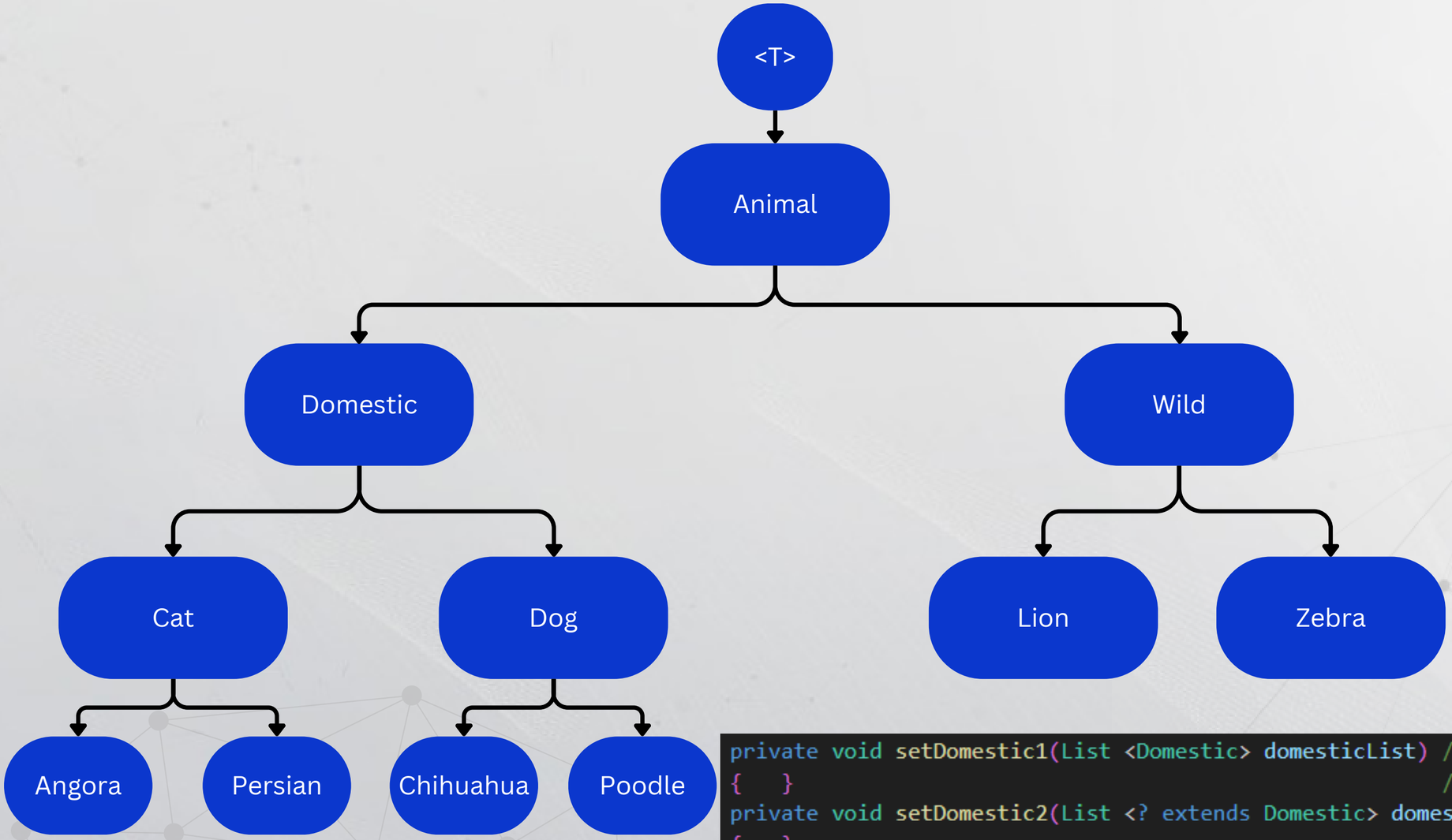
Let's see it running.

DAY 4
LESSON 3

Wildcards

Creative Devs
Software
MOBILE APPS•WEBSITES•AR•VIDEO GAMES•SECURITY

GENERICS AND WILDCARDS



Let's see the code.

```
private void setDomestic1(List <Domestic> domesticList) // Generics
{   }                                                   // List with Domestic
private void setDomestic2(List <? extends Domestic> domesticList) // Generecs & lower bound wildcards
{   }                                                             // List with Domestic or lower classes
private void setDomestic3(List <? super Domestic> domesticList) // Generics & upper bound wildcards
{   }                                                           // List with Domestic or upper classes
```

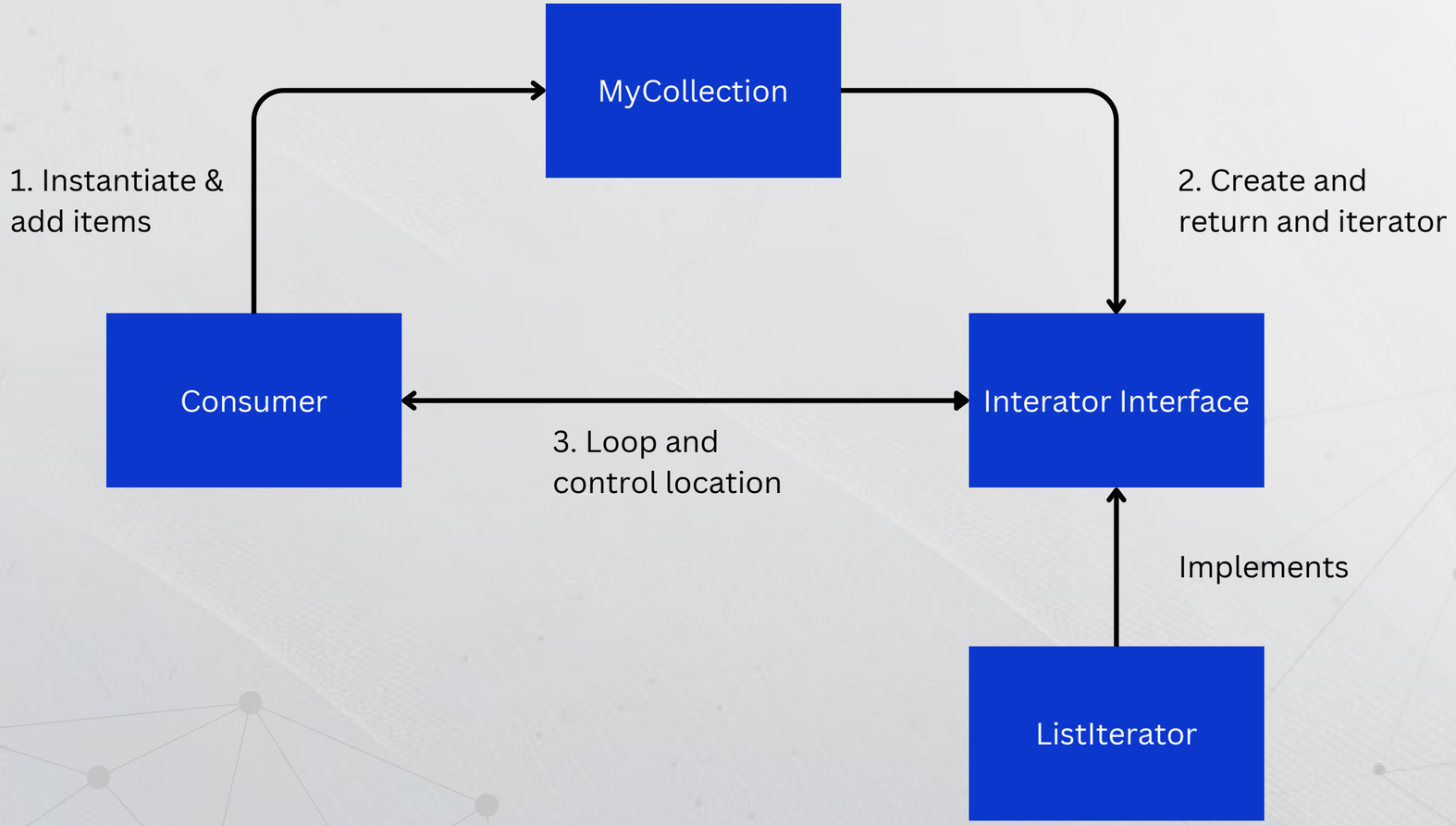
DAY 4
LESSON 4

Iterator Design Pattern

Creative Devs.
Software/>
MOBILE APPS•WEBSITES•AR•VIDEO GAMES•SECURITY

UNDERSTANDING DESIGN PATTERNS

Iterator Pattern



Let's Take a Look.

DAY 4
LESSON 5

Garbage Collection

Creative Devs
Software
MOBILE APPS•WEBSITES•AR•VIDEO GAMES•SECURITY

GARBAGE COLLECTION

Release objects you no longer need;

```
object = null;
```

Propose JVM perform garbage collection

```
System.gc();
```

Use appropriate memory management patterns

- **Weak references**
- Soft references
- Phantom references

```
Map<String, WeakReference<Image>> weakRefObject = new WeakHashMap<>();
```

```
Object strongRefObject = new Object();
WeakReference<Object> weakRef = new WeakReference<>(strongRefObject);
```

Let's see the code.

CODING TRIVA QUESTION

Change items type from String to Boolean and make it display;

How is now

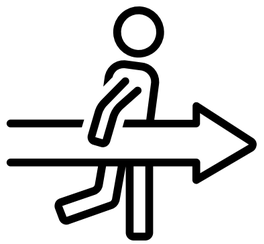
```
Item: Item 1
Item: Item 2
Item: Item 3
```

How it should be

```
Item: true
Item: true
Item: false
```

CODING TRIVA ANSWER

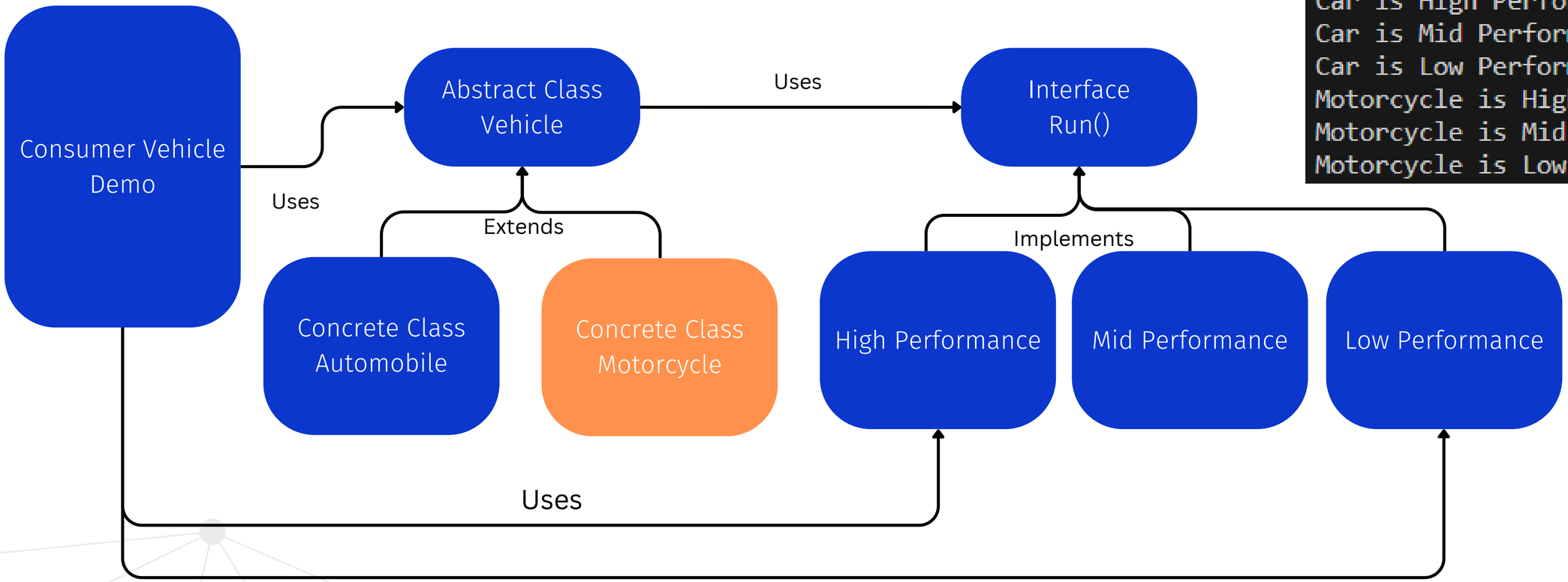
```java
package com.mains.Iterator;


public class IteratorPatternExample {
    Run | Debug
    public static void main(String[] args) {
        MyCollection<Boolean> collection = new MyCollection<>();
        collection.addItem(item:true);
        collection.addItem(item:true);
        collection.addItem(item:false);

        Iterator<Boolean> iterator = collection.createIterator();
        while (iterator.hasNext()) {
            Boolean item = iterator.next();
            System.out.println("Item: " + item);
        }
    }
}
```



Let's Take a Look.

CODING EXERCISE TO GO

Remember this, we added mid performance, now add Motorcycle Vehicle type in order to print what's shown on the right.




```
Car is High Performance
Car is Mid Performance
Car is Low Performance
Motorcycle is High Performance
Motorcycle is Mid Performance
Motorcycle is Low Performance
```

THANK YOU

Crash Course

We will see you tomorrow