Ganesh Kalyan Kommisetti

**Java 8 Features**

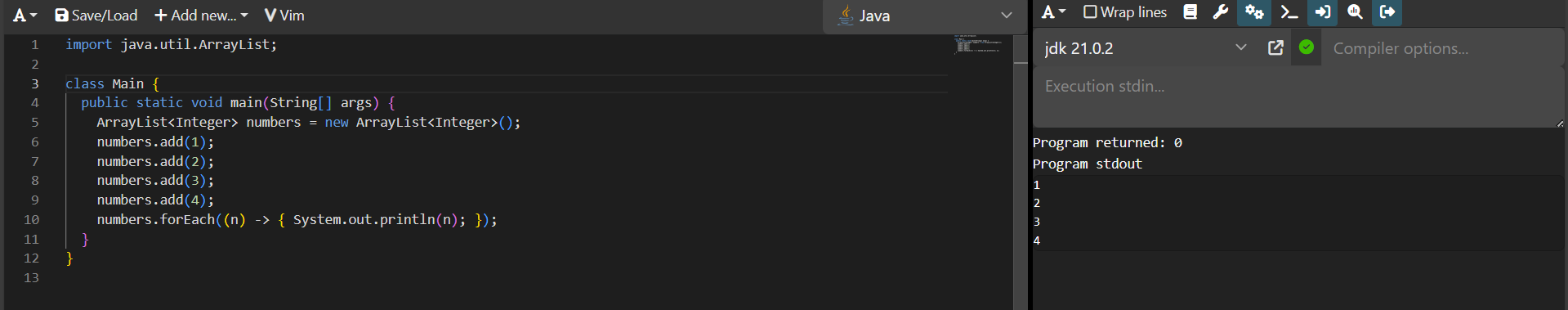
**Create a new branch feature-java8 as part of your github repo rg-assignments**

**and save/push all your coding solutions in the same branch**

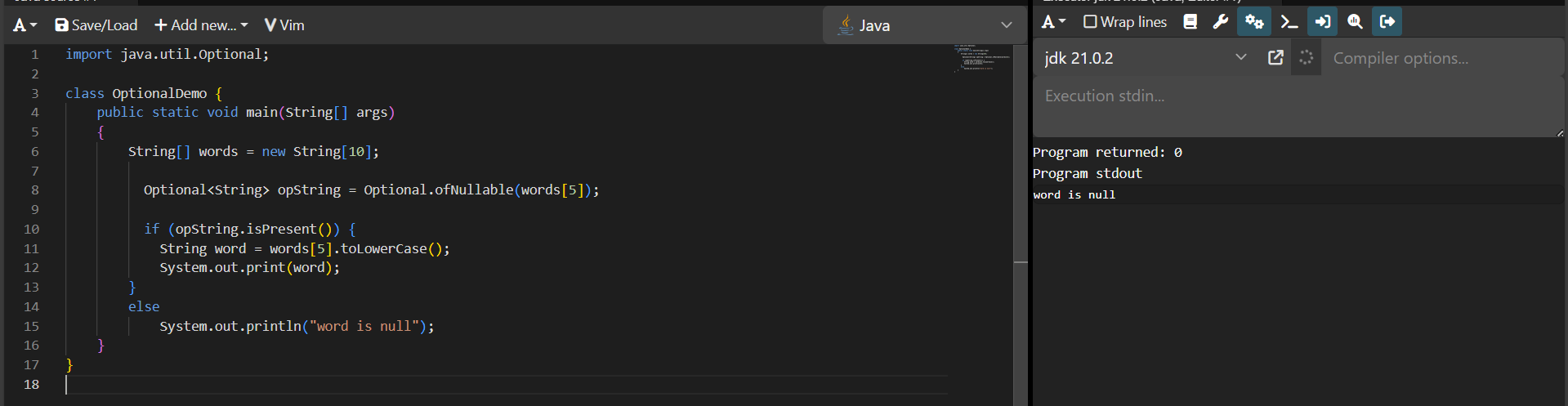
**Please share your output screenshots in the assignment document along with the github link for each question. Provide an explanation wherever possible as part of your response :-)**

**NOTE: Install Azul Zulu OpenJDK 8**

1. **List the features of Java 8**
2. Lambda expressions, stream apis, date/time api, default & static methods, Optional class etc.,
3. What is a Lambda Expression, and why do we use them? **Explain with a coding example and share the output screenshot.**
4. Short block of code which takes parameters and returns result.

****

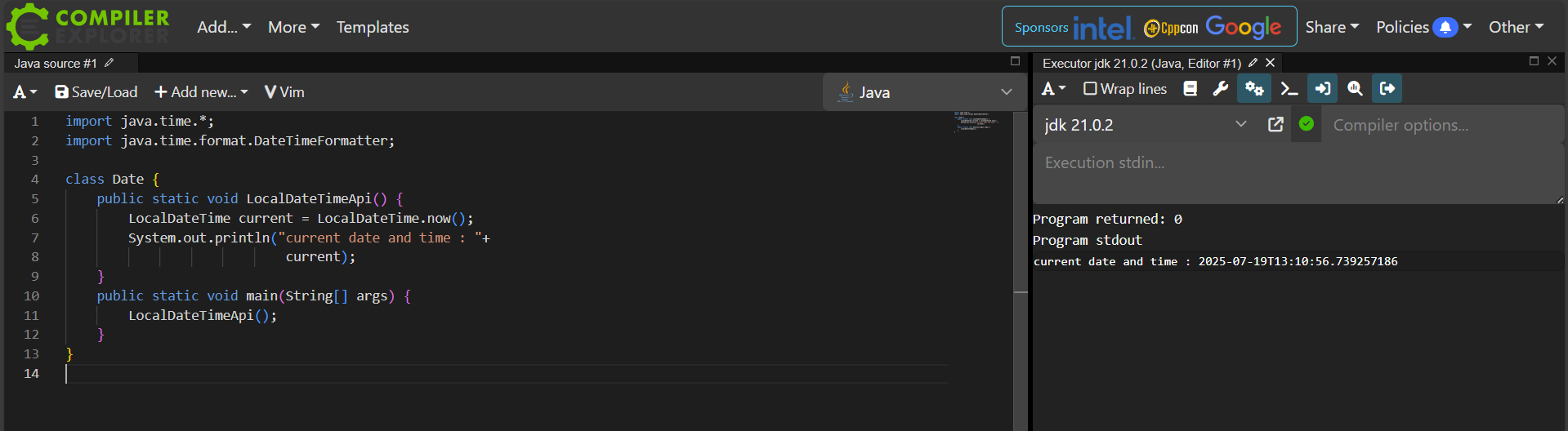
1. What is optional, and what is it best used for? **Explain with a coding example and share the output screenshot.**
2. A class in java that can be used to represent an item which may or may not be non null.



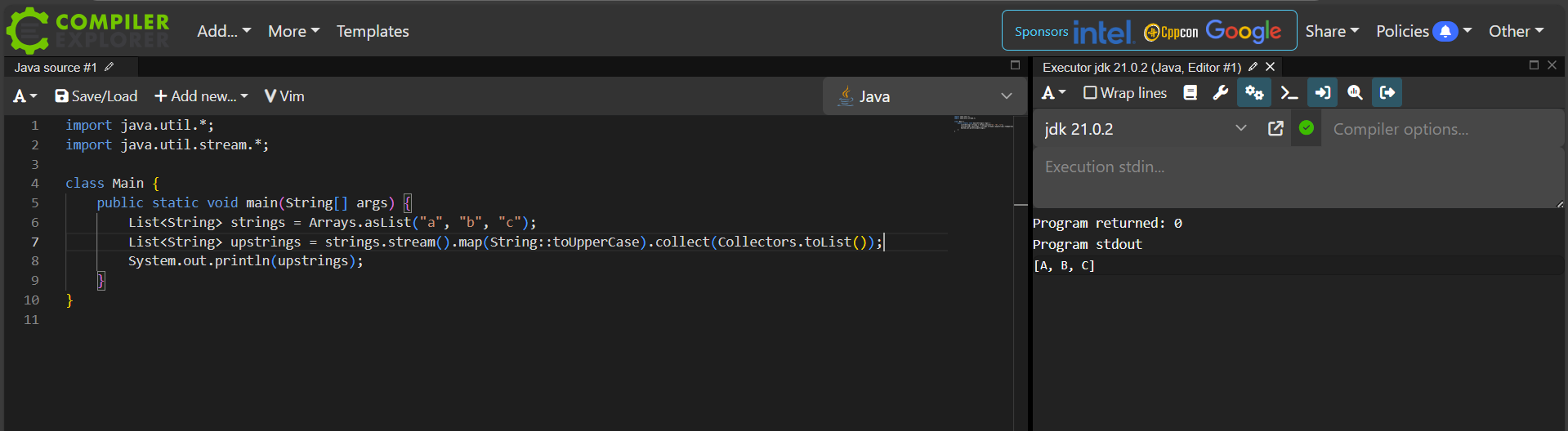
1. What is a functional interface? List some examples of predefined functional interfaces.
2. Interface which contains only one abstract method.

Examples include: Runnable, Callable, ActionListener etc.,

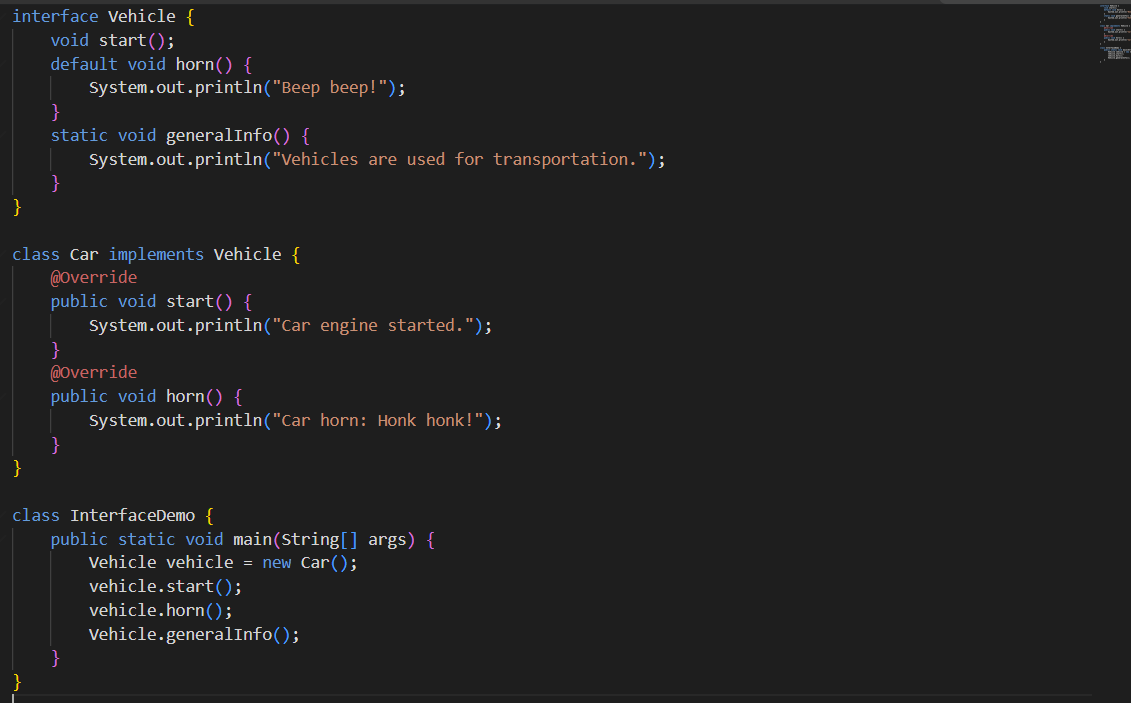
1. How are functional interfaces and Lambda Expressions related?
2. Lambda expressions can be used to implement method of functional interfaces.
3. List some Java 8 Date and Time API’s. **How will you get the current date and time using Java 8 Date and Time API? Write the implementation and share the output screenshot.**
4. java.time.LocalDate, java.time.LocalTime

****

1. **How to use map to convert objects into Uppercase in Java 8? Write the implementation and share the output screenshot.**
2. Use map, pass string toUpperCase function to it and collect the result.

****

1. Explain how Java 8 has enhanced interface functionality with default and static methods. **Why were these features introduced, explain with a coding example?**
2. Default methods are introduced to ensure backward compatibility and static methods help for having methods coupled with interfaces.



### Discuss the significance of the Stream API introduced in Java 8 for data processing. How does it improve application performance and developer productivity?

1. Stream API made management of arrays easy by allowing developers to write efficient and clean code in a clear and concise way. By using parallel streaming, application performance is improved by making use of multiple threads.

### What are method references in Java 8, and how do they complement the use of lambda expressions? Provide an example where a method reference is more suitable than a lambda expression. **Explain with a coding example and share the output screenshot.**

1. Used to make the shorthand notation of lambda functions when parameters and expected output are same.

# **Maven, Spring and Spring Boot**

**Create a new branch feature-spring as part of your github repo rg-assignments**

**and save/push all your code in the same branch**

**Please share your output screenshots in the assignment document along with the github link for each question. Provide an explanation wherever possible as part of your response :-)**

**NOTE: Install Spring version 5.3.28 and Spring Boot version 2.7.13**

|  |
| --- |
| 1. **Install maven 3.6 or above. Execute mvn -v in the local terminal/command prompt and share the screenshot**      1. **What is the difference between maven central repository and local repository?**   Maven central repository exists in the cloud and stores all the packages which can be accessed by everyone whereas local repository is stored in the users machine.   1. **Maven commands**    1. **To build the maven project**    2. **To run the maven tests**   mvn clean install  mvn test   1. **Please locate the maven settings.xml file and local maven repository in your machine and share the screenshot**     Haven’t made any configurational changes so no settings.xml file in my maven installation   1. **The basic principle behind Dependency Injection(DI) is that the objects define their dependencies .What are the different ways in which an object can define its dependency ?**   The main principle behind DI is not to create dependency objects every time they are used but get them from somewhere else.  Ways are: Through construction injections, autowiring, setter injection   1. **What is the difference between the @Autowired and @Inject annotation?**   @Inject is a language level annotation provided by java wheread @Autowired is specific to spring.   1. **Explain the use of @Respository, @Component, @Service and @Controller annotations with an example for each.**   @Repository – marks a class as Data access object  @Component – generic stereotypic for classes managed by spring container  @Controller – defines a web controller class  @Service – marks a class as business logic layer   1. **Fix the code and explain why?**   **The following code tries to inject a property from application.properties, but the appName field is always null. Identify and fix the issue.**  **@Component**  **public class AppNamePrinter {**  **@Value("app.name")**  **private String appName;**    **public void printAppName() {  System.out.println("Application Name: " + appName);  }  }**  @Component  public class AppNamePrinter {  @Value("${app.name}")  private String appName;    public void printAppName() {  System.out.println("Application Name: " + appName);  }  }   1. **What does the @SpringBootApplication annotation do?**   Marks a class as Spring Boot Application and enables configuration & component scan   1. **What is the maven command to start the SpringBootApplication?**   mvn spring-boot:run   1. **Implement EmployeeCRUD using Spring and JDBC with the below Employee class. In the branch feature-spring, create a folder Employee-Spring. Push the solution to the branch and share the link.**   **class Employee{**  **private int id;**  **private String name;**  **private String department;**  **}** |

1. **Implement EmployeeCRUD using SpringBoot and Spring Data JPA with the below Employee class. In the branch feature-spring, create a folder Employee-SpringBoot-JPA. Push the solution to the branch and share the link.**

**class Employee{**

**private int id;**

**private String name;**

**private String department;**

**}**

1. **Follow the demo in the pre-work link** [**https://www.youtube.com/watch?v=hr2XTbKSdAQ&t=18s**](https://www.youtube.com/watch?v=hr2XTbKSdAQ&t=18s) **and create a Spring Batch application that processes customer data. In the branch feature-spring, create a folder Customer-SpringBatch. Push the solution to the branch and share the link.**