Java Full Stack Developer - 500 Interview Questions & Answers with Examples

1. Q: What is a lambda expression in Java 8? (1)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How does CompletableFuture work in Java 8? (2)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (3)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (4)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (5)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is a lambda expression in Java 8? (6)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain Stream API in Java 8 with an example. (7)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (8)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain Stream API in Java 8 with an example. (9)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (10)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (11)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (12)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the use of @Autowired in Spring? (13)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you use method references in Java 8? (14)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (15)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the use of @Autowired in Spring? (16)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How does CompletableFuture work in Java 8? (17)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (18)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you use method references in Java 8? (19)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (20)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain Stream API in Java 8 with an example. (21)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (22)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (23)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How does CompletableFuture work in Java 8? (24)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (25)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain Stream API in Java 8 with an example. (26)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (27)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (28)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain the difference between HashMap and Hashtable. (29)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How does CompletableFuture work in Java 8? (30)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (31)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (32)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you define a REST endpoint in Spring Boot? (33)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How does CompletableFuture work in Java 8? (34)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (35)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (36)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you use method references in Java 8? (37)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (38)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (39)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (40)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (41)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (42)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (43)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you use method references in Java 8? (44)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (45)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain the difference between HashMap and Hashtable. (46)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you use method references in Java 8? (47)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (48)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain Stream API in Java 8 with an example. (49)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (50)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (51)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (52)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (53)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is a lambda expression in Java 8? (54)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (55)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is a lambda expression in Java 8? (56)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (57)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you use method references in Java 8? (58)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (59)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the use of @Autowired in Spring? (60)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the use of @Autowired in Spring? (61)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain Stream API in Java 8 with an example. (62)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (63)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is a lambda expression in Java 8? (64)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is a lambda expression in Java 8? (65)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is a lambda expression in Java 8? (66)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How does CompletableFuture work in Java 8? (67)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (68)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (69)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is a lambda expression in Java 8? (70)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain Stream API in Java 8 with an example. (71)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (72)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the use of @Autowired in Spring? (73)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you use method references in Java 8? (74)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (75)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain the difference between HashMap and Hashtable. (76)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the use of @Autowired in Spring? (77)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain Stream API in Java 8 with an example. (78)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (79)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain the difference between HashMap and Hashtable. (80)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the purpose of Optional in Java 8? (81)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the use of @Autowired in Spring? (82)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How does CompletableFuture work in Java 8? (83)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is a lambda expression in Java 8? (84)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How does CompletableFuture work in Java 8? (85)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (86)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (87)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (88)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (89)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (90)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is the purpose of Optional in Java 8? (91)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the use of @Autowired in Spring? (92)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain the difference between HashMap and Hashtable. (93)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain the difference between HashMap and Hashtable. (94)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain the difference between HashMap and Hashtable. (95)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you define a REST endpoint in Spring Boot? (96)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain Stream API in Java 8 with an example. (97)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (98)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you use method references in Java 8? (99)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (100)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is a lambda expression in Java 8? (101)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is the use of @Autowired in Spring? (102)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain the difference between HashMap and Hashtable. (103)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain Stream API in Java 8 with an example. (104)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (105)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (106)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the use of @Autowired in Spring? (107)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How does CompletableFuture work in Java 8? (108)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (109)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How does CompletableFuture work in Java 8? (110)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you use method references in Java 8? (111)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (112)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the use of @Autowired in Spring? (113)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the purpose of Optional in Java 8? (114)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (115)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (116)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (117)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (118)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain Stream API in Java 8 with an example. (119)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (120)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (121)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain the difference between HashMap and Hashtable. (122)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you use method references in Java 8? (123)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (124)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (125)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the purpose of Optional in Java 8? (126)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (127)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (128)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (129)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain the difference between HashMap and Hashtable. (130)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain Stream API in Java 8 with an example. (131)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (132)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you use method references in Java 8? (133)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (134)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (135)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How do you use method references in Java 8? (136)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you use method references in Java 8? (137)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (138)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How does CompletableFuture work in Java 8? (139)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is a lambda expression in Java 8? (140)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain Stream API in Java 8 with an example. (141)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (142)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain Stream API in Java 8 with an example. (143)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (144)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (145)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain Stream API in Java 8 with an example. (146)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (147)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (148)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is a lambda expression in Java 8? (149)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is a lambda expression in Java 8? (150)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (151)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (152)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (153)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the use of @Autowired in Spring? (154)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is a lambda expression in Java 8? (155)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you use method references in Java 8? (156)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (157)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain Stream API in Java 8 with an example. (158)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (159)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (160)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (161)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you define a REST endpoint in Spring Boot? (162)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain the difference between HashMap and Hashtable. (163)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain the difference between HashMap and Hashtable. (164)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain Stream API in Java 8 with an example. (165)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (166)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you define a REST endpoint in Spring Boot? (167)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is a lambda expression in Java 8? (168)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is the use of @Autowired in Spring? (169)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is a lambda expression in Java 8? (170)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is the use of @Autowired in Spring? (171)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain the difference between HashMap and Hashtable. (172)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you use method references in Java 8? (173)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you use method references in Java 8? (174)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (175)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (176)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you use method references in Java 8? (177)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (178)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the use of @Autowired in Spring? (179)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is a lambda expression in Java 8? (180)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is the purpose of Optional in Java 8? (181)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the use of @Autowired in Spring? (182)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the purpose of Optional in Java 8? (183)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (184)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (185)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How do you use method references in Java 8? (186)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (187)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (188)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the purpose of Optional in Java 8? (189)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the use of @Autowired in Spring? (190)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the use of @Autowired in Spring? (191)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain the difference between HashMap and Hashtable. (192)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain Stream API in Java 8 with an example. (193)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (194)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How does CompletableFuture work in Java 8? (195)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (196)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain the difference between HashMap and Hashtable. (197)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you define a REST endpoint in Spring Boot? (198)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is a lambda expression in Java 8? (199)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you use method references in Java 8? (200)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (201)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (202)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (203)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you use method references in Java 8? (204)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (205)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is a lambda expression in Java 8? (206)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain Stream API in Java 8 with an example. (207)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you use method references in Java 8? (208)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (209)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain Stream API in Java 8 with an example. (210)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you use method references in Java 8? (211)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (212)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (213)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (214)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the use of @Autowired in Spring? (215)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the use of @Autowired in Spring? (216)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How does CompletableFuture work in Java 8? (217)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the use of @Autowired in Spring? (218)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is a lambda expression in Java 8? (219)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How does CompletableFuture work in Java 8? (220)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the use of @Autowired in Spring? (221)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you define a REST endpoint in Spring Boot? (222)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How does CompletableFuture work in Java 8? (223)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is a lambda expression in Java 8? (224)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (225)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How does CompletableFuture work in Java 8? (226)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you use method references in Java 8? (227)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (228)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (229)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain the difference between HashMap and Hashtable. (230)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you define a REST endpoint in Spring Boot? (231)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the use of @Autowired in Spring? (232)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the use of @Autowired in Spring? (233)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you use method references in Java 8? (234)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (235)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the use of @Autowired in Spring? (236)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How does CompletableFuture work in Java 8? (237)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (238)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (239)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How do you use method references in Java 8? (240)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (241)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain the difference between HashMap and Hashtable. (242)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you define a REST endpoint in Spring Boot? (243)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain the difference between HashMap and Hashtable. (244)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the purpose of Optional in Java 8? (245)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (246)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain Stream API in Java 8 with an example. (247)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (248)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How does CompletableFuture work in Java 8? (249)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (250)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain Stream API in Java 8 with an example. (251)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (252)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you define a REST endpoint in Spring Boot? (253)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is a lambda expression in Java 8? (254)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is the purpose of Optional in Java 8? (255)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How do you use method references in Java 8? (256)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (257)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (258)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (259)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How do you use method references in Java 8? (260)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (261)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain the difference between HashMap and Hashtable. (262)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How does CompletableFuture work in Java 8? (263)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (264)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (265)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (266)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you define a REST endpoint in Spring Boot? (267)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (268)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (269)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How does CompletableFuture work in Java 8? (270)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the use of @Autowired in Spring? (271)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the use of @Autowired in Spring? (272)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you define a REST endpoint in Spring Boot? (273)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you define a REST endpoint in Spring Boot? (274)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain Stream API in Java 8 with an example. (275)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (276)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (277)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the use of @Autowired in Spring? (278)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you use method references in Java 8? (279)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (280)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you define a REST endpoint in Spring Boot? (281)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain the difference between HashMap and Hashtable. (282)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the use of @Autowired in Spring? (283)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain Stream API in Java 8 with an example. (284)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (285)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (286)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you use method references in Java 8? (287)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (288)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you use method references in Java 8? (289)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (290)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the use of @Autowired in Spring? (291)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you use method references in Java 8? (292)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (293)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain Stream API in Java 8 with an example. (294)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (295)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you use method references in Java 8? (296)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (297)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is the purpose of Optional in Java 8? (298)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (299)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (300)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain the difference between HashMap and Hashtable. (301)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How does CompletableFuture work in Java 8? (302)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (303)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (304)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (305)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (306)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain Stream API in Java 8 with an example. (307)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you use method references in Java 8? (308)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (309)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is a lambda expression in Java 8? (310)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you use method references in Java 8? (311)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (312)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How does CompletableFuture work in Java 8? (313)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the use of @Autowired in Spring? (314)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the purpose of Optional in Java 8? (315)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (316)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (317)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain the difference between HashMap and Hashtable. (318)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you use method references in Java 8? (319)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (320)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is a lambda expression in Java 8? (321)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (322)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you use method references in Java 8? (323)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (324)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is a lambda expression in Java 8? (325)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (326)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (327)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the use of @Autowired in Spring? (328)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is a lambda expression in Java 8? (329)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is the purpose of Optional in Java 8? (330)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (331)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (332)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (333)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you define a REST endpoint in Spring Boot? (334)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (335)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (336)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (337)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (338)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the use of @Autowired in Spring? (339)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is the use of @Autowired in Spring? (340)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain Stream API in Java 8 with an example. (341)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (342)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the use of @Autowired in Spring? (343)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you define a REST endpoint in Spring Boot? (344)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is a lambda expression in Java 8? (345)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you use method references in Java 8? (346)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (347)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (348)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (349)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (350)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is a lambda expression in Java 8? (351)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is the purpose of Optional in Java 8? (352)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (353)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (354)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How does CompletableFuture work in Java 8? (355)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (356)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is a lambda expression in Java 8? (357)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (358)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How does CompletableFuture work in Java 8? (359)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (360)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you define a REST endpoint in Spring Boot? (361)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain the difference between HashMap and Hashtable. (362)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you define a REST endpoint in Spring Boot? (363)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is a lambda expression in Java 8? (364)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is a lambda expression in Java 8? (365)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain Stream API in Java 8 with an example. (366)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (367)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (368)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (369)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is a lambda expression in Java 8? (370)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain Stream API in Java 8 with an example. (371)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (372)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (373)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is a lambda expression in Java 8? (374)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is a lambda expression in Java 8? (375)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How does CompletableFuture work in Java 8? (376)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (377)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you define a REST endpoint in Spring Boot? (378)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How does CompletableFuture work in Java 8? (379)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you use method references in Java 8? (380)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (381)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (382)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (383)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you use method references in Java 8? (384)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (385)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How does CompletableFuture work in Java 8? (386)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (387)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How does CompletableFuture work in Java 8? (388)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (389)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the use of @Autowired in Spring? (390)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you use method references in Java 8? (391)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (392)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the purpose of Optional in Java 8? (393)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (394)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you use method references in Java 8? (395)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (396)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain Stream API in Java 8 with an example. (397)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (398)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain the difference between HashMap and Hashtable. (399)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How does CompletableFuture work in Java 8? (400)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (401)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (402)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (403)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain Stream API in Java 8 with an example. (404)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (405)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the purpose of Optional in Java 8? (406)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (407)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the use of @Autowired in Spring? (408)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you use method references in Java 8? (409)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (410)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain the difference between HashMap and Hashtable. (411)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How does CompletableFuture work in Java 8? (412)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (413)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is a lambda expression in Java 8? (414)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you use method references in Java 8? (415)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (416)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (417)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (418)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain Stream API in Java 8 with an example. (419)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (420)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (421)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (422)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: Explain Stream API in Java 8 with an example. (423)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (424)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you use method references in Java 8? (425)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (426)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain Stream API in Java 8 with an example. (427)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (428)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain the difference between HashMap and Hashtable. (429)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you define a REST endpoint in Spring Boot? (430)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (431)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (432)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is a lambda expression in Java 8? (433)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (434)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain Stream API in Java 8 with an example. (435)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (436)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the use of @Autowired in Spring? (437)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is a lambda expression in Java 8? (438)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is a lambda expression in Java 8? (439)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (440)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the use of @Autowired in Spring? (441)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: Explain the difference between HashMap and Hashtable. (442)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How does CompletableFuture work in Java 8? (443)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (444)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain the difference between HashMap and Hashtable. (445)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: What is the purpose of Optional in Java 8? (446)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (447)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is a lambda expression in Java 8? (448)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is a lambda expression in Java 8? (449)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is the purpose of Optional in Java 8? (450)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (451)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain the difference between HashMap and Hashtable. (452)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How do you define a REST endpoint in Spring Boot? (453)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (454)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (455)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (456)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: Explain the difference between HashMap and Hashtable. (457)

A: HashMap is not synchronized and permits null keys/values. Hashtable is synchronized and does not allow nulls.

Map<String, String> map = new HashMap<>();  
map.put(null, "value");

1. Q: How does CompletableFuture work in Java 8? (458)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (459)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (460)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you use method references in Java 8? (461)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (462)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you use method references in Java 8? (463)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (464)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (465)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (466)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How do you use method references in Java 8? (467)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (468)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you define a REST endpoint in Spring Boot? (469)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you use method references in Java 8? (470)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (471)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How do you define a REST endpoint in Spring Boot? (472)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (473)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: Explain Stream API in Java 8 with an example. (474)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is a lambda expression in Java 8? (475)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: What is the use of @Autowired in Spring? (476)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How does CompletableFuture work in Java 8? (477)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (478)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is the use of @Autowired in Spring? (479)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is a lambda expression in Java 8? (480)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain Stream API in Java 8 with an example. (481)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (482)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (483)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you define a REST endpoint in Spring Boot? (484)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How do you define a REST endpoint in Spring Boot? (485)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: What is the purpose of Optional in Java 8? (486)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: What is a lambda expression in Java 8? (487)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: Explain Stream API in Java 8 with an example. (488)

A: Stream API is used to process collections of objects in a functional style.

List<String> names = Arrays.asList("Alice", "Bob", "Charlie");  
names.stream().filter(name -> name.startsWith("A")).forEach(System.out::println);

1. Q: How does CompletableFuture work in Java 8? (489)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: How do you define a REST endpoint in Spring Boot? (490)

A: Use @RestController and @RequestMapping annotations.

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String sayHello() {  
 return "Hello World";  
 }  
}

1. Q: How does CompletableFuture work in Java 8? (491)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);

1. Q: What is the purpose of Optional in Java 8? (492)

A: Optional is a container object which may or may not contain a non-null value.

Optional<String> name = Optional.ofNullable("John");  
name.ifPresent(System.out::println);

1. Q: How do you use method references in Java 8? (493)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you use method references in Java 8? (494)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: How do you use method references in Java 8? (495)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (496)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: How do you use method references in Java 8? (497)

A: Method reference is a shorthand notation of a lambda expression to call a method.

List<String> list = Arrays.asList("a", "b", "c");  
list.forEach(System.out::println);

1. Q: What is the use of @Autowired in Spring? (498)

A: It is used to automatically inject beans.

public class Service {  
 @Autowired  
 private Repository repo;  
}

1. Q: What is a lambda expression in Java 8? (499)

A: A lambda expression is a short block of code which takes in parameters and returns a value.

(a, b) -> a + b

1. Q: How does CompletableFuture work in Java 8? (500)

A: It allows you to write asynchronous code in a more readable way.

CompletableFuture.supplyAsync(() -> "Hello")  
 .thenApply(result -> result + " World")  
 .thenAccept(System.out::println);