**Interview Questions**

**Java Platform**

1 . Why is Java so popular?

2 . What is platform independence?

3 . What is bytecode?

4 . Compare JDK vs JVM vs JRE

5 . What are the important differences between C++ and Java?

6 . What is the role for a classloader in Java?

**Wrapper Classes**

7 . What are Wrapper classes?

8 . Why do we need Wrapper classes in Java?

9 . What are the different ways of creating Wrapper class instances?

10 . What are differences in the two ways of creating Wrapper classes?

11 . What is auto boxing?

12 . What are the advantages of auto boxing?

13 . What is casting?

14 . What is implicit casting?

15 . What is explicit casting?

**Strings**

16 . Are all String’s immutable?

17 . Where are String values stored in memory?

18 . Why should you be careful about String concatenation(+) operator in loops?

19 . How do you solve above problem?

20 . What are differences between String and StringBuffer?

21 . What are differences between StringBuilder and StringBuffer?

22 . Can you give examples of different utility methods in String class?

**Object oriented programming basics**

23 . What is a class?

24 . What is an object?

25 . What is state of an object?

26 . What is behavior of an object?

27 . What is the super class of every class in Java?

28 . Explain about toString method ?

29 . What is the use of equals method in Java?

30 . What are the important things to consider when implementing equals method?

31 . What is the Hashcode method used for in Java?

32 . Explain inheritance with examples .

33 . What is method overloading?

34 . What is method overriding?

35 . Can super class reference variable can hold an object of sub class?

36 . Is multiple inheritance allowed in Java?

37 . What is an interface?

38 . How do you define an interface?

39 . How do you implement an interface?

40 . Can you explain a few tricky things about interfaces?

41 . Can you extend an interface?

42 . Can a class extend multiple interfaces?

43 . What is an abstract class?

44 . When do you use an abstract class?

45 . How do you define an abstract method?

46 . Compare abstract class vs interface?

47 . What is a constructor?

48 . What is a default constructor?

49 . Will this code compile?

50 . How do you call a super class constructor from a constructor?

51 . Will this code compile?

52 . What is the use of this()?

53 . Can a constructor be called directly from a method?

54 . Is a super class constructor called even when there is no explicit call from a sub class constructor?

**Advanced object oriented concepts**

55 . What is polymorphism?

56 . What is the use of instanceof operator in Java?

57 . What is coupling?

58 . What is cohesion?

59 . What is encapsulation?

60 . What is an inner class?

61 . What is a static inner class?

62 . Can you create an inner class inside a method?

63 . What is an anonymous class?

**Modifiers**

64 . What is default class modifier?

65 . What is private access modifier?

66 . What is default or package access modifier?

67 . What is protected access modifier?

68 . What is public access modifier?

69 . What access types of variables can be accessed from a class in same package?

70 . What access types of variables can be accessed from a class in different package?

71 . What access types of variables can be accessed from a sub class in same package?

72 . What access types of variables can be accessed from a sub class in different package?

73 . What is the use of a final modifier on a class?

74 . What is the use of a final modifier on a method?

75 . What is a final variable?

76 . What is a final argument?

77 . What happens when a variable is marked as volatile?

78 . What is a static variable?

**conditions & loops**

79 . Why should you always use blocks around if statement?

80 . Guess the output

81 . Guess the output

82 . Guess the output of this switch block .

83 . Guess the output of this switch block?

84 . Should default be the last case in a switch statement?

85 . Can a switch statement be used around a String

86 . Guess the output of this for loop

87 . What is an enhanced for loop?

88 . What is the output of the for loop below?

89 . What is the output of the program below?

90 . What is the output of the program below?

**Exception handling**

91 . Why is exception handling important?

92 . What design pattern is used to implement exception handling features in most languages?

93 . What is the need for finally block?

94 . In what scenarios is code in finally not executed?

95 . Will finally be executed in the program below?

96 . Is try without a catch is allowed?

97 . Is try without catch and finally allowed?

98 . Can you explain the hierarchy of exception handling classes?

99 . What is the difference between error and exception?

100 . What is the difference between checked exceptions and unchecked exceptions?

101 . How do you throw an exception from a method?

102 . What happens when you throw a checked exception from a method?

103 . What are the options you have to eliminate compilation errors when handling checked exceptions?

104 . How do you create a custom exception?

105 . How do you handle multiple exception types with same exception handling block?

106 . Can you explain about try with resources?

107 . How does try with resources work?

108 . Can you explain a few exception handling best practices?

**Miscellaneous topics**

109 . What are the default values in an array?

110 . How do you loop around an array using enhanced for loop?

111 . How do you print the content of an array?

112 . How do you compare two arrays?

113 . What is an enum?

114 . Can you use a switch statement around an enum?

115 . What are variable arguments or varargs?

116 . What are asserts used for?

117 . When should asserts be used?

118 . What is garbage collection?

119 . Can you explain garbage collection with an example?

120 . When is garbage collection run?

121 . What are best practices on garbage collection?

122 . What are initialization blocks?

123 . What is a static initializer?

124 . What is an instance initializer block?

125 . What is tokenizing?

126 . Can you give an example of tokenizing?

127 . What is serialization?

128 . How do you serialize an object using serializable interface?

129 . How do you de-serialize in Java?

130 . What do you do if only parts of the object have to be serialized?

131 . How do you serialize a hierarchy of objects?

132 . Are the constructors in an object invoked when it is de-serialized?

133 . Are the values of static variables stored when an object is serialized?

**Collections**

134 . Why do we need collections in Java?

135 . What are the important interfaces in the collection hierarchy?

136 . What are the important methods that are declared in the collection interface?

137 . Can you explain briefly about the List interface?

138 . Explain about ArrayList with an example?

139 . Can an ArrayList have duplicate elements?

140 . How do you iterate around an ArrayList using iterator?

141 . How do you sort an ArrayList?

142 . How do you sort elements in an ArrayList using comparable interface?

143 . How do you sort elements in an ArrayList using comparator interface?

144 . What is vector class? How is it different from an ArrayList?

145 . What is linkedList? What interfaces does it implement? How is it different from an ArrayList?

146 . Can you briefly explain about the Set interface?

147 . What are the important interfaces related to the Set interface?

148 . What is the difference between Set and sortedSet interfaces?

149 . Can you give examples of classes that implement the Set interface?

150 . What is a HashSet?

151 . What is a linkedHashSet? How is different from a HashSet?

152 . What is a TreeSet? How is different from a HashSet?

153 . Can you give examples of implementations of navigableSet?

154 . Explain briefly about Queue interface?

155 . What are the important interfaces related to the Queue interface?

156 . Explain about the Deque interface?

157 . Explain the BlockingQueue interface?

158 . What is a priorityQueue?

159 . Can you give example implementations of the BlockingQueue interface?

160 . Can you briefly explain about the Map interface?

161 . What is difference between Map and sortedMap?

162 . What is a HashMap?

163 . What are the different methods in a Hash Map?

164 . What is a TreeMap? How is different from a HashMap?

165 . Can you give an example of implementation of navigableMap interface?

166 . What are the static methods present in the collections class?

**Advanced collections**

167 . What is the difference between synchronized and concurrent collections in Java?

168 . Explain about the new concurrent collections in Java?

169 . Explain about copyonwrite concurrent collections approach?

170 . What is compareandswap approach?

171 . What is a lock? How is it different from using synchronized approach?

172 . What is initial capacity of a Java collection?

173 . What is load factor?

174 . When does a Java collection throw UnsupportedOperationException?

175 . What is difference between fail-safe and fail-fast iterators?

176 . What are atomic operations in Java?

177 . What is BlockingQueue in Java?

**Generics**

178 . What are Generics?

179 . Why do we need Generics? Can you give an example of how Generics make a program more flexible?

180 . How do you declare a generic class?

181 . What are the restrictions in using generic type that is declared in a class declaration?

182 . How can we restrict Generics to a subclass of particular class?

183 . How can we restrict Generics to a super class of particular class?

184 . Can you give an example of a generic method?

**Multi threading**

185 . What is the need for threads in Java?

186 . How do you create a thread?

187 . How do you create a thread by extending thread class?

188 . How do you create a thread by implementing runnable interface?

189 . How do you run a thread in Java?

190 . What are the different states of a thread?

191 . What is priority of a thread? How do you change the priority of a thread?

192 . What is executorservice?

193 . Can you give an example for executorservice?

194 . Explain different ways of creating executor services .

195 . How do you check whether an executionservice task executed successfully?

196 . What is callable? How do you execute a callable from executionservice?

197 . What is synchronization of threads?

198 . Can you give an example of a synchronized block?

199 . Can a static method be synchronized?

200 . What is the use of join method in threads?

201 . Describe a few other important methods in threads?

202 . What is a deadlock?

203 . What are the important methods in Java for inter-thread communication?

204 . What is the use of wait method?

205 . What is the use of notify method?

206 . What is the use of notifyall method?

207 . Can you write a synchronized program with wait and notify methods?

**Functional Programming - Lamdba expressions and Streams**

208 . What is functional programming?

209 . Can you give an example of functional programming?

210 . What is a stream?

211 . Explain about streams with an example?

what are intermediate operations in streams?

212 . What are terminal operations in streams?

213 . What are method references?

214 . What are lambda expressions?

215 . Can you give an example of lambda expression?

216 . Can you explain the relationship between lambda expression and functional interfaces?

217 . What is a predicate?

218 . What is the functional interface - function?

219 . What is a consumer?

220 . Can you give examples of functional interfaces with multiple arguments?

**Java 8**

1. What are new features which got introduced in Java 8?

* Lambda Expression
* Default methods in interface
* Functional Interface
* Optional
* Method references
* Date API
* Stream API
* Nashorn, JavaScript Engine

1. What are main advantages of using Java 8?

* More compact code (compact= กระชับ, กะทัดรัด)
* Less boilerplate code (โค้ดน้อยลง)
* More readable and reusable code (โค๊ดอ่านง่ายขึ้นและนำไปใช้ใหม่ได้)
* More testable code (โค๊ดทดสอบได้มากขึ้น)
* Parallel operations (ดำเนินงานแบบคู่ขนาน)

1. What is Optional? Why and how can you use it?

Optional คือคลาสใหม่ใน Java 8 ที่บรรจุค่าที่เป็นตัวเลือกซึ่งเป็นได้ทั้งมีค่าหรือไม่มีค่าก็ได้

**Optional** is a new class in Java 8 that encapsulates an optional value which is either present or not.

Optional ตัวบรรจุ object อย่างหนึ่งซึ่งอาจมีค่าหรือไม่มีค่าที่ไม่ใช่ null

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

เราสามารถใช้ตัวเลือกเพื่อหลีกเลี่ยง NullPointerException ใน Java โดยใช้ isPresent เพื่อตรวจสอบว่ามีค่าภายในอ็อพเจ็กต์ตัวเลือกหรือไม่

We can use Optional to avoid NullPointerException in Java with using ***isPresent*** to check if there is a value inside the Optional object.

isPresent คือ API ของคลาสที่เป็นตัวเลือกซึ่งใช้ในการตรวจสอบว่ามีค่าอยู่ภายในอ็อพเจ็กต์ตัวเลือกหรือไม่

The isPresent is an API of an Optional class which used to check if there is a value inside the Optional object.

These methods which belong to an Optional class as follows.

**Optional.empty**: We can create an empty Optional by using the Optional.empty() method and will return an empty Optional instance which a value is not present for this Optional.

เราสามารถสร้างตัวเลือกที่ว่างเปล่าโดยใช้เมธอด Optional.empty () และจะคืนค่าอินสแตนซ์ตัวเลือกที่ว่างเปล่าซึ่งไม่มีค่าสำหรับตัวเลือกนี้

**Optional.ofNullable**:A method allows creating an optional that can accept null.

**Optional.of:** We can create an optional with a specified value using Optional.of()

**Optonal.get:** It is a method to obtain a value present in an Optional instance returns the value if a value is present in this Optional, otherwise throws NoSuchElementException if there is no value present.

วิธีการเพื่อทำให้ได้ค่าที่มีอยู่ในอินสแตนซ์ตัวเลือก จะส่งกลับค่าถ้าค่ามีอยู่ในตัวเลือกนี้ มิฉะนั้นจะโยน NoSuchElementException หากไม่มีค่าอยู่

**Optonal.orElse:** This method returns the value if present otherwise returns the default value provided as a parameter or return other.

เมธอดนี้จะส่งค่ากลับถ้ามี มิฉะนั้นจะส่งกลับค่าดีฟอลต์ที่ระบุเป็นพารามิเตอร์หรือส่งกลับค่าอื่นๆ

**Optonal.orElse:** This method returns the value if present, otherwise throw an exception created by the provided supplier.

เมธอดนี้จะส่งค่ากลับถ้ามี มิฉะนั้นจะโยนความผิดผลาดที่ได้จัดเตรียมไว้

1. What is lambda expression?

* Lambda expression is a new feature which is introduced in Java 8. A lambda expression is an anonymous function that doesn’t have a name and doesn’t belong to any class. Lambda Expression can accept a set of input parameters and returns results but it can also be defined as an Anonymous Function that allows users to pass methods as arguments.

1. What is Nashorn in Java8?

* Nashorn is the new Javascript processing engine for the Java platform which added in Java 8.