---------------------------------------------------------------------------------------------------

Question 1: What is the difference between an abstract class and an interface in Java 8?

Answer:

* Abstract class can have constructors whereas interface cannot have a constructor.
* A class can extend only one abstract class , but can implement multiple interfaces. Thus, a class can inherit multiple properties from multiple sources only through interfaces, not through abstract classes.

---------------------------------------------------------------------------------------------------

Question 2: In the Java contract between equals() and hashCode(), can

2 non Equal objects have the same Hashcode?

2 objects with different Hashcodes be equal?

Answer:

- > No, as per the hashcode and equals contract .

- > yes the two unequal may have the same hashcode

---------------------------------------------------------------------------------------------------

Question 3: What is the use of hashcode method?

Answer:

Its main purpose is to facilitate hashing in hash tables, which are used by data structures like HashMap.

---------------------------------------------------------------------------------------------------

Question 4: Given 2 threads running in parallel, one Producing objects and another Consuming said objects,

how do you pass objects from the first to the second Thread?

Answer: we can use a blocking queue to communicate between these two threads.

---------------------------------------------------------------------------------------------------

Question 5: What is an Immutable Object? How do you make objects of a class immutable?

Answer: An object is considered immutable if its state cannot change after it is constructed.

By marking the state of an object to final will make the object of a class immutable.

---------------------------------------------------------------------------------------------------

Question 6: What is the type T in the following expression?

T expression = Object::hashcode;

Answer:

The T denotes a input type of the functional interface which uses the Function<Object,Boolean> interface for comparision

---------------------------------------------------------------------------------------------------

Question 7: 7. Complete the following method to print the value of optionalValue when present

public void print(Optional<String> optionalValue) {

}

Answer: optionalValue.ifPresent(s -> System.out.println(s));

------------------------------------------------------------------------------------------------------

Question 8: What is wrong with the following method?

public Date getNowIfAbsent(Optional<Date> optionalDate) {

return optionalDate.orElse(new Date()) ; }

Answer:

Nothing wrong it looks fine

------------------------------------------------------------------------------------------------------

Question 9: Simplify the following method:

public boolean containsCaseInsensitive(@Nonnull List<String> available, @Nonnull String wanted) {

return available.stream()

.filter(Objects::nonNull)

.map(String::toLowerCase)

.filter(s -> s.equals(wanted.toLowerCase()))

.findFirst()

.map(s -> true).orElse(false);

}

Answer:

return available.stream()

.map(String::toLowerCase)

.filter(s -> s.equals(wanted.toLowerCase()))

.findFirst() .forEach(s -> true);

------------------------------------------------------------------------------------------------------