1. Date/Time API (JSR 310)

1. Which of the following will correctly create a `LocalDate` instance representing July 30, 2024?

```
LocalDate date = LocalDate.of(2024, 7, 30);
```

a) `LocalDate date = LocalDate.create(2024, 7, 30);`
b) `LocalDate date = LocalDate.of(2024, 30, 7);`
c) `LocalDate date = LocalDate.of(7, 30, 2024);`
d) `LocalDate date = LocalDate.of(2024, 7, 30);`

2. How can you get the day of the week from a `LocalDate` object?

```
LocalDate date = LocalDate.of(2024, 7, 30);
DayOfWeek dayOfWeek = date.getDayOfWeek();
```

a) `date.getDay();`
b) `date.getWeekDay();`
c) `date.getDayOfWeek();`
d) `date.getDayOfWeekValue();`

3. What is the output of the following code snippet?

```
LocalDateTime dateTime = LocalDateTime.parse("2024-07-30T15:30:00");
String formattedDateTime = dateTime.format(DateTimeFormatter.ofPattern("yyyy/MM/dd
HH:mm:ss"));
System.out.println(formattedDateTime);
```

a) `2024-07-30 15:30:00`
b) `2024/07/30 15:30:00`
c) `2024/07/30 15:30`
d) `2024/07/30 15:30:00`

2. Generic Classes

4. Which of the following correctly uses a bounded wildcard in a generic class?

```
public void processItems(List<? extends Number> list)
{
        // method implementation
```

```
    }
```

a) `public void processItems(List<? super Integer> list)`
b) `public void processItems(List<? extends Integer> list)`
c) `public void processItems(List<? extends Number> list)`
d) `public void processItems(List<?> list)`

5. What is the result of the following code snippet?

```
List<?> list = new ArrayList<String>();
list.add("test");
```

a) Compiles and runs successfully
b) Compilation error
c) List accepts any type and adds "test" successfully
d) List accepts String and adds "test" successfully

3. Collections Framework

6. What will be the output of the following code snippet?

```
List<String> list = new ArrayList<>(Arrays.asList("apple", "banana", "cherry"));
Collections.sort(list, Comparator.reverseOrder());
System.out.println(list);
```

a) `[apple, banana, cherry]`
b) `[cherry, banana, apple]`
c) `[banana, cherry, apple]`
d) `[banana, apple, cherry]`

7. Which of the following is a thread-safe collection?

a) `ArrayList`
b) `LinkedList`
c) `Vector`
d) `HashSet`

8. What does the `Stream.map()` method do?

a) Filters the elements of the stream
b) Transforms each element of the stream
c) Sorts the elements of the stream
d) Reduces the elements of the stream

9. What is the output of the following code snippet?

```
Stream<String> stream = Stream.of("a", "b", "c");
String result = stream.reduce("", (a, b) -> a + b);
System.out.println(result);
```

a) `abc`
b) `a`
c) `b`
d) `c`

4. Working with Stacks

10. What is the output of the following code snippet?

```
Stack<Integer> stack = new Stack<>();
          stack.push(1);
          stack.push(2);
          stack.push(3);
          System.out.println(stack.pop());
          System.out.println(stack.peek());
```

a) `3 2`
b) `3 1`
c) `2 3`
d) `3 3`

11. How can you implement a stack using a linked list?

a) By using an `ArrayList`
b) By using a `LinkedList` and utilizing its methods
c) By using a `HashMap`
d) By using an `ArrayDeque`

5. Working with Queues

12. What is the output of the following code snippet using a circular queue?

```
Queue<Integer> queue = new ArrayDeque<>(3);
          queue.add(1);
          queue.add(2);
          queue.add(3);
          queue.poll();
          queue.add(4);
          System.out.println(queue);
```

a) `[2, 3, 4]`
b) `[1, 2, 4]`
c) `[2, 3, 1]`
d) `[2, 4, 3]`

13. Which method is used to remove an element from a queue in ?

a) `remove()`
b) `dequeue()`
c) `pop()`
d) `poll()`

6. Sorting & Searching Algorithms

14. What is the time complexity of Quick Sort in the average case?

a) O(n log n)
b) O(n^2)
c) O(n)
d) O(log n)

15. What is the worst-case time complexity of Bubble Sort?

a) O(n log n)
b) O(n^2)
c) O(n)
d) O(log n)

16. What is the result of the following code snippet using Selection Sort?

```
int[] array = { 64, 25, 12, 22, 11 };
        for (int i = 0; i < array.length - 1; i++) {
            int minIndex = i;
            for (int j = i + 1; j < array.length;
j++) {
                if (array[j] < array[minIndex]) {
                    minIndex = j;
                }
            }
            int temp = array[minIndex];
            array[minIndex] = array[i];
            array[i] = temp;
        }

      System.out.println(Arrays.toString(array));
```

a) `[11, 12, 22, 25, 64]`
b) `[64, 25, 22, 12, 11]`
c) `[11, 22, 12, 25, 64]`
d) `[11, 12, 22, 25, 64]`

7. Input & Output Streams

17. How do you write a string to a file using `FileWriter`?

```
FileWriter writer = new FileWriter("output.txt");
writer.write("Hello, World!");
writer.close();
```

a) Correct code
b) `FileWriter` cannot be used to write strings
c) `write` method requires `BufferedWriter`
d) Code will not compile

18. What is the purpose of `Serializable` in ?

a) To serialize objects into a byte stream
b) To serialize objects into a JSON format
c) To store objects in a database
d) To compress object data

8. Multi-Threading

19. What is the correct way to start a thread using the `Runnable` interface?

```
Runnable task = () -> System.out.println("Task is running");
Thread thread = new Thread(task);
thread.start();
```

a) Correct code
b) `Thread` cannot be started with `Runnable`
c) `Runnable` should extend `Thread`
d) `Runnable` requires a `start` method

20. What is the purpose of `synchronized` in ?

a) To synchronize the system clock
b) To ensure that only one thread accesses a block of code at a time

c) To create multiple threads simultaneously
d) To prioritize thread execution

9. JDBC API

21. How do you execute a SQL query using `PreparedStatement`?

```
String query = "SELECT * FROM users WHERE age > ?";
PreparedStatement pstmt = connection.prepareStatement(query);
pstmt.setInt(1, 25);
ResultSet rs = pstmt.executeQuery();
```

a) Correct code
b) `PreparedStatement` cannot execute queries
c) `setInt` method is incorrect
d) `executeQuery` method should be called directly on `Statement`

22. What is the purpose of `ResultSetMetaData` in JDBC?

a) To fetch the results of a query
b) To provide information about the types and properties of the columns in a `ResultSet`
c) To modify data in the database
d) To create database tables

10. Performing Unit Testing using JUnit4

23. What is the annotation used to specify a test method in JUnit4?

a) `@Test`
b) `@TestMethod`
c) `@TestCase`
d) `@RunTest`

24. Which annotation is used to execute code before each test method?

a) `@Before`
b) `@BeforeClass`
c) `@After`
d) `@AfterClass`

25. How do you write a parameterized test in JUnit4?

```
@RunWith(Parameterized.class)
    public class ExampleTest {
        @Parameterized.Parameters
        public static Collection<Object[]> data() {
            return Arrays.asList(new Object[][] {
```

```
                { 1, 2 },
                { 3, 4 }
            });
        }
        private int input1;
        private int input2;

        public ExampleTest(int input1, int input2) {
            this.input1 = input1;
            this.input2 = input2;
        }

        @Test
        public void test() {
            assertEquals(input1 + input2, 5);  //
Example test logic
        }
    }
```

a) Correct code
b) `@Parameters` should be `@TestParameters`
c) No parameterized tests in JUnit4
d) `@RunWith` is not required