Started on	Monday, 28 April 2025, 3:17 PM
State	Finished
Completed on	Monday, 28 April 2025, 3:26 PM
Time taken	9 mins 35 secs
Marks	7.00/10.00
Grade	**70.00** out of 100.00

Question 1

Complete

Mark 1.00 out of 1.00

ExecutorService service = Executors.newFixedThreadPool(2);

Future<Integer> future1 = service.submit(() -> 1);

Future<Integer> future2 = service.submit(() -> 2);

System.out.print(future1.isDone() + " " + future2.isDone());

service.shutdown();

At the point of printing, what is most likely?

○ a.   true false

◉ b.   Any combination depending on timing

○ c.   false false

○ d.   true true

Question 2

Complete

Mark 0.00 out of 1.00

What is wrong with this Runnable usage?

Runnable r = () -> {

   Thread.sleep(1000);

   System.out.println("Done");

};

new Thread(r).start();

○ a.   Thread.sleep must be inside try-catch block

◉ b.   Multiple threads will be created

○ c.   No problem

○ d.   Compile-time error due to missing return

Question 3

Complete

Mark 0.00 out of 1.00

What happens for the following code?

Callable<String> c = () -> {

    Thread.sleep(2000);

    return "Result";

};

ExecutorService executor = Executors.newFixedThreadPool(1);

Future<String> future = executor.submit(c);

System.out.print(future.get(1, TimeUnit.SECONDS));

-   a.   TimeoutException
-   b.   IllegalStateException
-   c.   ExecutionException
- ◉ d.   "Result" will be printed

Question 4

Complete

Mark 1.00 out of 1.00

What will happen when the following code is executed?

class MyCallable implements Callable<Integer> {

    public Integer call() {

        return 100;

    }

}

public class Test {

    public static void main(String[] args) {

        ExecutorService service = Executors.newSingleThreadExecutor();

        Future<Integer> future = service.submit(new MyCallable());

        service.shutdown();

    }

}

-   a.   call() will never be executed because get() is missing.
-   b.   Compile-time error.
-   c.   call() will throw an exception.
- ◉ d.   call() will still be executed even without future.get().

Question 5

Complete

Mark 1.00 out of 1.00

What will be the output of the following code?

```java
class MyRunnable implements Runnable {
    public void run() {
        System.out.print("Runnable");
    }
}
public class Test {
    public static void main(String[] args) {
        Thread t = new Thread(new MyRunnable());
        t.start();
    }
}
```

- ○ a.   Runnable
- ○ b.   Compile-time error
- ○ c.   No output
- ○ d.   Runtime exception

Question 6

Complete

Mark 1.00 out of 1.00

In the following code, how many threads are created?

```java
public class Test {
    public static void main(String[] args) {
        ExecutorService service = Executors.newFixedThreadPool(5);
        for (int i = 0; i < 10; i++) {
            service.submit(() -> System.out.print(Thread.currentThread().getName() + " "));
        }
        service.shutdown();
    }
}
```

- ○ a.   5
- ○ b.   10
- ○ c.   Depends on JVM
- ○ d.   15

Question 7

Complete

Mark 0.00 out of 1.00

In this code, what type does the Future hold?

Future<?> future = executor.submit(() -> System.out.println("Task"));

- ○ a. Future<String>
- ○ b. Future<Object>
- ○ c. Future<Integer>
- ○ d. Future<Void>

Question 8

Complete

Mark 1.00 out of 1.00

Identify the problem in the following code:

```
class MyTask implements Runnable {
    public String run() {
        return "Hello";
    }
}
```

- ○ a. Valid code
- ○ b. Infinite loop
- ○ c. Runtime exception
- ◉ d. Compile-time error because Runnable.run() must return void

Question 9

Complete

Mark 1.00 out of 1.00

What will be the output?

class MyCallable implements Callable<String> {

    public String call() {

        return "Callable";

    }

}

public class Test {

    public static void main(String[] args) throws Exception {

        FutureTask<String> task = new FutureTask<>(new MyCallable());

        new Thread(task).start();

        System.out.print(task.get());

    }

}

- ○ a.   Runtime exception
- ○ b.   null
- ◉ c.   Callable
- ○ d.   Compile-time error

Question 10

Complete

Mark 1.00 out of 1.00

What happens in this code?

ExecutorService executor = Executors.newSingleThreadExecutor();

Future<String> future = executor.submit(() -> {

    throw new RuntimeException("Error occurred!");

});

future.get();

- ○ a.   Future returns null
- ○ b.   Thread terminates silently
- ◉ c.   ExecutionException is thrown when get() is called
- ○ d.   No Exception is thrown