```
1.  class MyClass<T, U>
2.  {
3.      T t;
4.      U u;
5.      MyClass(T t, U u)
6.      {
7.          this.t = t;
8.          this.u = u;
9.      }
10.       public void print()
11.     {
12.         System.out.println(t);
13.         System.out.println(u);
14.     }
15. }
16. class Main
17. {
18.     public static void main (String[] args)
19.     {
20.         MyClass <String, Integer> obj =
21.             new MyClass<String, Integer>("Aarushi", 15);
22.
23.         obj.print();
24.     }
25. }
```

Output:

Aarushi

15

It will call print method then, String is taken as t and Integer is taken as u.

Q 8 What will be the output of the following code?

```java
import java.util.EnumSet;
enum Test
{
    A, B, C, D, E
};
public class Example
{
    public static void main(String[] args)
    {

        EnumSet<Test> a1, a2, a3, a4;

        a1 = EnumSet.of(Test.D, Test.C, Test.B, Test.A);
        a2 = EnumSet.complementOf(a1);
        a3 = EnumSet.allOf(Test.class);
        a4 = EnumSet.range(Test.A, Test.C);
        System.out.println("Set 1: " + a1);
        System.out.println("Set 2: " + a2);
        System.out.println("Set 3: " + a3);
        System.out.println("Set 4: " + a4);
    }
}
```

Output:

Set 1: [D,C,B,A]

Set 2: [E]

Set 3: [A,B,C,D,E]

Set 4: [A,B,C]

Of : Creates an enum set initially containing the specified element.

Complement.Of: Creates an enum set with the same element type as the specified enum set initially it contains the elements which do not exist in the specified element.

AllOf: Creates an enum set containing all of the elements in the specified element type.

Range: Creates an enum set initially containing all of the elements in the range defined by the two specified endpoints.

Q 10 In the below piece of code, method go() will throw an exception. The developer has used three catch blocks. To which of the following class types would the thrown exception belongs and what will be the output of the code?

```
1.  class MainClass {
2.   public static void main(String[] args) {
3.    try {
4.     go();
5.    } catch (Exception e) {
6.     System.out.println("1");
7.    } catch (Error e){
8.     System.out.println("2");
9.    } catch (Throwable t){
10.    System.out.println("3");
11.   }
12.  }
13.  static void go(){
14.   go();
15.  }
16. }
```

Output:

Error,2

As due to infinite recursion it will full up the stack due to which compiler will show stack Overflow Error.

Q 14 There are 5 threads in the waiting pool of a monitor 'mon' and all these threads have the same priority. One of the threads is thread1. How can you notify thread1, so that it alone moves from Waiting state to Ready state?

Ops:
A. ○ Execute thread1.notify(); from any code(synchronized or not) of any objects
B. ○ You cannot specify which thread will get notified
C. ○ Execute thread1.notify(); from a synchronized code of any objects
D. ◉ Execute mon.notify(thread1); from a synchronized code of any objects

Answer:

You cannot specify which thread will get notified.

Q 15 What should be done if an event listener has to perform a lengthy task i.e. checking spelling in a large document and would not be able to process additional UI events till the task is completed, that makes the program appear to freeze?

Ops: A. The event listener should perform the UI events on the foremost priority

B. The event listener should hand off long tasks to another thread

C. The event listener should cancel the lengthy task if it is not complete in a specific time

D. The event listener should prioritize the tasks

Output:

The event listener should hand off long tasks to another thread.

Multithreading will help into multitasking and prevent the program to freeze.

Q 17 What will be the output of the following code?

```
1.  class TestJoinMethod2 extends Thread{
2.   public void run(){
3.    for(int i=1;i<=3;i++){
4.     try{
5.      Thread.sleep(500);
6.     }catch(Exception e){System.out.println(e);}
7.    System.out.print(i);
8.    }
9.   }
10. public static void main(String args[]){
11.  TestJoinMethod2 t1=new TestJoinMethod2();
12.  TestJoinMethod2 t2=new TestJoinMethod2();
13.  TestJoinMethod2 t3=new TestJoinMethod2();
14.  t1.start();
15.  try{
16.   t1.join(1500);
17.  }catch(Exception e){System.out.print(e);}
18.
19.  t2.start();
20.  t3.start();
21.  }
22. }
```

Output:

1

2

3

1

1

2

2

3

3

Q 16 What will be the output of the following code?

```
1.
2.  public class Test implements Runnable{
3.  public static void main(String[] args){
4.  Thread t = new Thread(this);
5.  t.start();
6.  }
7.
8.  public void run(){
9.  System.out.println("test");
10. }
11. }
```

Output:

The program does not compile because this cannot be referenced in a static method.