**zx306 Ziyue Xu OOP Supo 1**

**Question 2:**

https://chime.cl.cam.ac.uk/page/repos/zx306/fibonacci/summary

1. The provided test only test the case when parameter i = -1 and i = 0. However, the other base cases where i = 1 / 2 and the general case where i > 2 are not tested. Thus we cannot ensure that the program runs correctly with those inputs. So we need more tests.
2. For small values of i, the difference of time taken between 2 functions can be negligible. Thus it’s hard to determine whether FibonacciTable makes use of cache or not.

**Question 5**

https://github.com/MaggieXuzy/OOP-Supo/blob/master/Supo1/src/TailRecursion.java

**Question7**

Variables of primitive type: d, f

Variables of references type: i[]. t, c

Classes: LinkedList<Double>

Objects: l, k

**Question 8**

https://chime.cl.cam.ac.uk/page/repos/zx306/classic\_collections\_lists\_and\_queues/summary

1. The empty list is represented with the head points to null.
2. The toString() method in class LinkList will check if the head is null or not. If it’s null, then it just returns “”. Otherwise, it will call the toString() method for each node. If next of the node is not null, it will concatenate value with “,”, and then concatenate next, which recursively calls toString() to get the value until next is null, where the recursion stops.
3. The static factory method can create an instance in different ways according to different inputs thus it’s easier to use and implement if there are several cases when creating an object.

|  |
| --- |
| LinkList |
| - head : Node |
| + create(int[] elements) : LinkList  - addFirst(int element): void  + toString(): String |

|  |
| --- |
| Node |
| - value: int  - next: Node |
| + toString(): String |

**Questions 10**

Test() should be a constructor thus it should have no return value. However Test() is defined with return type void so it’s not a constructor. It’s just a function and will not effect the value of variable outside the function.

**Question 14**

1. https://github.com/MaggieXuzy/OOP-Supo/blob/master/Supo1/src/Vector2D.java
2. To make it immutable, the attributes will be private instead of public. final will be added. So it’s declared as private final float x;
3. (1) For the mutable version, the vector v just adds to the value of this vector. The vector value changes. For the immutable version, this can’t be done as the attributes can’t be changed.
4. This works for both mutable and immutable versions as a new Vector2D is returned. The attributes’ value don’t change in this case.
5. This also works for both and the attributes don’t change for both vectors v1 and v2. A new Vector2D is created and returned.
6. The static method means that no instance is needed to be created to call this method. The method can be called with Vector2D.add(v1, v2). It also works for both versions.
7. The values of variables cannot be changed or assessed by the user.

**Question 15**

The parameterised type should be an object and primitive types like int or double is not an object. Integer or Double can be used as parameterised type. When running, Java doesn’t know what t type actually is so T() is forbidden.

**Question 20**

