CS2040S Tutorial 1

AY 24/25 Sem 2 — github/omgeta

Q1. (a.) A class is a blueprint for specific instances/object which share similar interfaces and fields. ■

(b.) `main` can be called without needing to instantiate an object of the class. ■

(c.) Example:

```
public class Box {
    private static x = 3;
}

public class Main {
    public static void main(String[] args) {
      System.out.println(Box.x) // Accessing a private variable
    outside of the class
    }
}
```

(d.) Interfaces are used to define an abstract public interface for similar classes. Example:

```
public interface Runnable {
    void run();
}

public class Car implements Runnable {
    public void run() {
      // implementation here
    }
}

public class WashingMachine implements Runnable {
    public void run() {
      // implementation here
    }
}
```

Yes; we can return objects with an interface type.

(e.) The final value of j will be 8 but the final value of i, k will still be 7. In `addOne`, the *int* value is passed by value and any changes do not actually affect the original value. In `myOtherIntAddOne`, k is only a variable holding a reference to the original and reassignment only reassigns where the variable points to and does the change the original k. ■

(f.) Yes, but the parameter name will shadow the unqualified member name. To still access the member/static variable, use a qualified name like `this.x` (for member) or `Main.x` (for static) ■

Q2. (a.) $O(n^3)$ ■

(b.) $O(n^2 \log n)$ ■

(c.) $O(n^5)$ ■

(d.) $O(2^{n^2})$ ■

Q3. (a.) $O(n)$ ■

(b.) $O(n \log n)$ ■

(c.) $O(\log n)$ ■

(d.) $O(n^{\log n})$ ■

Q4. Naive solution: iterate through each value in the array checking if the next increments by 1
Fast solution: use binary search comparing value at *mid* with its expected index choosing left or right appropriately ■

Q5. Use binary search with the initial values of *low* and *high* being 1 and the pile taking the most time. At each step, find *mid* and check if its possible to finish within h hours using a helper function `isFeasible(piles, k, h)`. If it is possible, set $low = mid$ else $high = mid$ and continue iterating until *low* and *high* converge on a single value which is the minimum time spent. ■

Q6. $O(n)$ solution: iterate through each point recording the maximum and minimum x, y values. At the end, the bounding box is simply
$(minX, minY), (minX, maxY), (maxX, maxY), (maxX, maxY)$ ■