CS2040S Tutorial 1

AY 24/25 Sem 2 — github/omgeta

Q1. (a.) Classes are blueprints for instances of objects, defining the general methods, attributes and behaviour of the objects.

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

(c.) Example:

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

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

(d.) Interfaces are used to define a contract followed by any class which implements it. This contract specifies the class must implement the methods specified by the interface.

```java
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.) $f_1(n) = 7.2 + 34n^3 + 3254n < 7.2n^3 + 34n^3 + 3254n^3 = O(n^3)$

(b.) $f_2(n) = n^2 \log n + 25n \log^2 n = O(n^2 \log n)$

(c.) $f_3(n) = 2^{4\log n} + 5n^5 = (2^{\log n})^4 + 5n^5 = n^4 + 5n^5 = O(n^5)$

(d.) $f_4(n) = 2^{2n^2+4n+7} = 2^{2n^2+4n} \cdot 2^7 = O(2^{2n^2+4n})$

Q3. (a.) $h_1(n) = f(n) + g(n) \leq c_1 n + c_2 \log n = O(n)$

(b.) $h_2(n) = f(n) \times g(n) \leq c_1 n \cdot c_2 \log n = c_1 c_2 n \log n = O(n \log n)$

(c.) $h_3(n) = \max(f(n), g(n)) = O(f(n) + g(n)) = O(n)$

(d.) $h_4(n) = f(g(n)) \leq c_1 g(n) \leq c_1 c_2 \log n = O(\log n)$

(e.) $h_5(n) = f(n)^{g(n)} = (c_1 n)^{c_2 \log n} = O(n^{c_2 \log n})$

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 value at the 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
$(minX, minY), (minX, maxY), (maxX, maxY), (maxX, maxY)$
$O(\log n)$ solution: use the $O(\log n)$ 1D peak finding algorithm 4 times to find the minimum and maximum x, y values.