

- I.
- C.
 - a.
 - C.
 - C.
 - d.
 - d.
 - b.
 - d.
 - C.
 - * b
- II.
- F.
 - F.
 - T.
 - T.
 - F.
 - F.
 - ~~T~~ F
 - T.
 - ~~T~~ F
 - T.

III.

1. Overriding = 繼承父類別的 method 之後，以相同名字、相同參數 (type, 數量)，相容的回值值，只改寫 body 內容。(存取權限只能相同或放大)

ex: `protected test() { ... }` → `public test() { // 不同內容 }`

Overloading = 在同一類別中，相同的 method 名，且參數一定要不同 (type, 數量)

ex:

```

public test (int x) { ... }
public test (double y) { ... }
public test (int a, int b) { ... }

```

2. `this()`: 呼叫此類別 (無參數) 的 constructor.
`super()`: 呼叫父類別 (無參數) 的 constructor.

3. `public`: 任何 class 都可存取,
`protected`: 同 package, 或不同 package 的子類別可存取.
`default`: 同 package 的 class 皆可存取.
`private`: 只有同 class 才可存取.

4. `class`: 一般的類別, 可實體化, 不可有 `abstract method`.
`abstract class`: 不可被實體化, 可同時有一般和抽象方法
`interface`: 其中的 variable 一定為 `public static final`,
method 不能有 body. 由之後 implements 此 interface 的 class 實作.

IV.

1. `t.run()` → `t.start()`. `myThread` ~~implements~~ `Thread` {
↓
`extends`

2. B.java:

```
public class B extends A, Object {  
    public B(String name) {  
        super(name);  
    }  
}
```

D.java:
↑ implements

```
public class D extends C {  
    public ← private int get { return }  
    public void set(int p) { }  
}
```

3. Scanner input = new Scanner (System.~~out~~);

~~try~~ {

p = input.nextInt();

~~} catch (IOException e) {~~

~~break;~~

~~}~~

4. ~~x.PI = 3.14;~~ → Sphere.PI = 3.14;

~~x.radius = 2;~~ → Sphere.radius = 2;

System.out.print("volume = " + ~~getvolume()~~);

↓
Sphere.getvolume()

▽

1. ~~reference~~

2. synchronization

3. Object

4. ~~if-else~~

5. ? :

6. 1.

7. (lock).wait()

8. notify() , notifyAll()

9. 0

VI

1. public class Node {

int num; // node 的编号

<Node>ArrayList edges; // node 連到 哪些其它 node.

public Node(int num) {

this.num = num; edges = new ArrayList();

}

public void addEdge(Node n) {

edges.add(n);

n.addEdge(this);

}

}

public class UndirGraph {

<Node>ArrayList nodes; // stores nodes of the graph.

public UndirGraph() {

nodes = new ArrayList();

}

public void addNode(Node n) {

nodes.add(n); // add the node to the list.

}

public void delNode(int num) {

// find the node with number num,

// and delete it.

}

}

```
2. public class ReverseSubString {  
    public static void main (String[] args) {  
        String s1 = args[2];  
        int start = Integer.parseInt (args[3]);  
        int end = Integer.parseInt (args[4]) + 1;  
        // 轉成 Int  
        s1 = args[2].substring (start, end);  
        char[] c = s1.toCharArray();  
        for (int i = c.length - 1; i >= 0; i--) {  
            System.out.print (c[i]);  
        }  
    }  
}
```