api/interface driven programming

[caller]

```
class A {
  void m1(int a, int b) {
    IB b = new BImpl();
    int y = b.m2(a);
    // logic
    ….
    ……
    ……….
  }
}
```

```
interface IB {
  int m2(int x);
}
```

[callee]

```
class BImpl implements IB {
  int m2(int x) {
    // logic
    return anyValue;
  }
}
```

api/interface-driven programming
1. The caller invokes/talks to the callee over fixed interface
2. unless the interface remains same, the caller can talk to any of the implementations of the interface
3. if there is change in interface itself then the caller will be impacted

so there are few dis-advantages or problems with interface/api driven programming
1. the components are tightly coupled through their interfaces
2. Always the classes will communicate with each other through their concrete references
3. within the Caller we write logic in invoking an specific interface of another concreate class.

event-driven programming model