Relaciona los conceptos con términos

- Clases 'Singleton'
- Alto acoplamiento
- Intestabilidad
- Optimización prematura
- Nombrado no descriptivo
- Duplicación

Respuesta 1

Principios STUPID que debemos evitar

```
interface IA {
        void F();
}
interface IB {
        void G();
}
interface IC {
        void H();
}
class A implements IA {
        IC c;
        public A(IC c) {
                this.c = c;
        }
        @Override
        public void F() {
        }
        public void F2() {
                c.J();
        }
}

class B implements IB {
        IA a;
        public B(IA a) {
                this.a = a;
        }
        @Override
        public void G() {
                a.F();
        }
}

class C implements IB {
        IB b;
        public C(IB b) {
                this.b = b;
        }
        @Override
        public void H() {
                b.G();
        }
        public void J() {
        }
}
```

Respuesta 2

Las clases están acopladas

- Principio de segregación de interfaz
- Principio de inversión de dependencias
- Principio abierto/cerrado
- Principio de resposabilidad única
- Principio de sustitución de Liskov

Respuesta 3

Principios SOLID que debemos seguir

```
class A {
  C c = new C();
        public void F() {
        }
        public void F2() {
                c.J();
        }
}

class B {
        A a = new A();
        public void G() {
                a.F();
        }
}

class C {
        B b = new B();
        public void H() {
                b.G();
        }
        public void J() {
        }
}
```

Relaciona los conceptos con términos

- Clases 'Singleton'
- Alto acoplamiento
- Intestabilidad
- Optimización prematura
- Nombrado no descriptivo
- Duplicación

Respuesta 1

Principios STUPID que debemos evitar

```
interface IA {
        void F();
}
interface IB {
        void G();
}
interface IC {
        void H();
}
class A implements IA {
        IC c;
        public A(IC c) {
                this.c = c;
        }
        @Override
        public void F() {
        }
        public void F2() {
                c.J();
        }
}

class B implements IB {
        IA a;
        public B(IA a) {
                this.a = a;
        }
        @Override
        public void G() {
                a.F();
        }
}

class C implements IB {
        IB b;
        public C(IB b) {
                this.b = b;
        }
        @Override
        public void H() {
                b.G();
        }
        public void J() {
        }
}
```

Respuesta 2

Las clases están acopladas

- Principio de segregación de interfaz
- Principio de inversión de dependencias
- Principio abierto/cerrado
- Principio de resposabilidad única
- Principio de sustitución de Liskov

Respuesta 3

Principios SOLID que debemos seguir

```
class A {
  C c = new C();
        public void F() {
        }
        public void F2() {
                c.J();
        }
}

class B {
        A a = new A();
        public void G() {
                a.F();
        }
}

class C {
        B b = new B();
        public void H() {
                b.G();
        }
        public void J() {
        }
}
```