

# How to build a stable system

Main concerns for data intensive app.

By: **Marcel Adel**



# Main concerns for data intensive app

01

## Reliability

The application works as it should.

02

## Scalability

System ability to cope with increased load

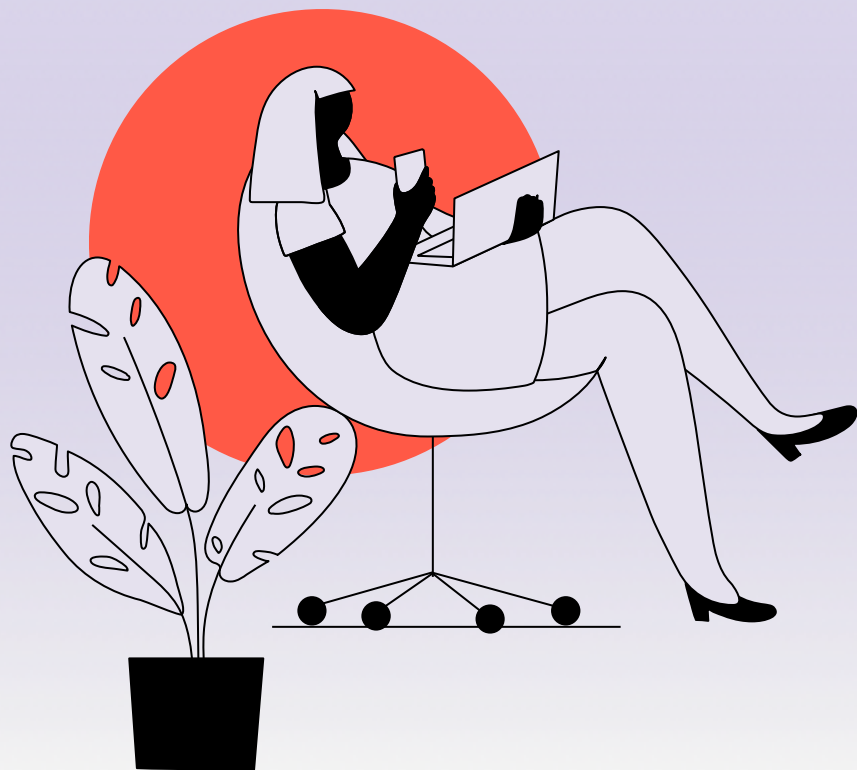
03

## Maintainability

Adapt to change



# Reliability



- The application works as expected.
- It should tolerate users making mistakes.
- Perform well under the expected load and data volume.
- System eligible if it is fault tolerant.



# Scalability



- The System should deal with increase in data , traffic volume.
- Cope with increased load.



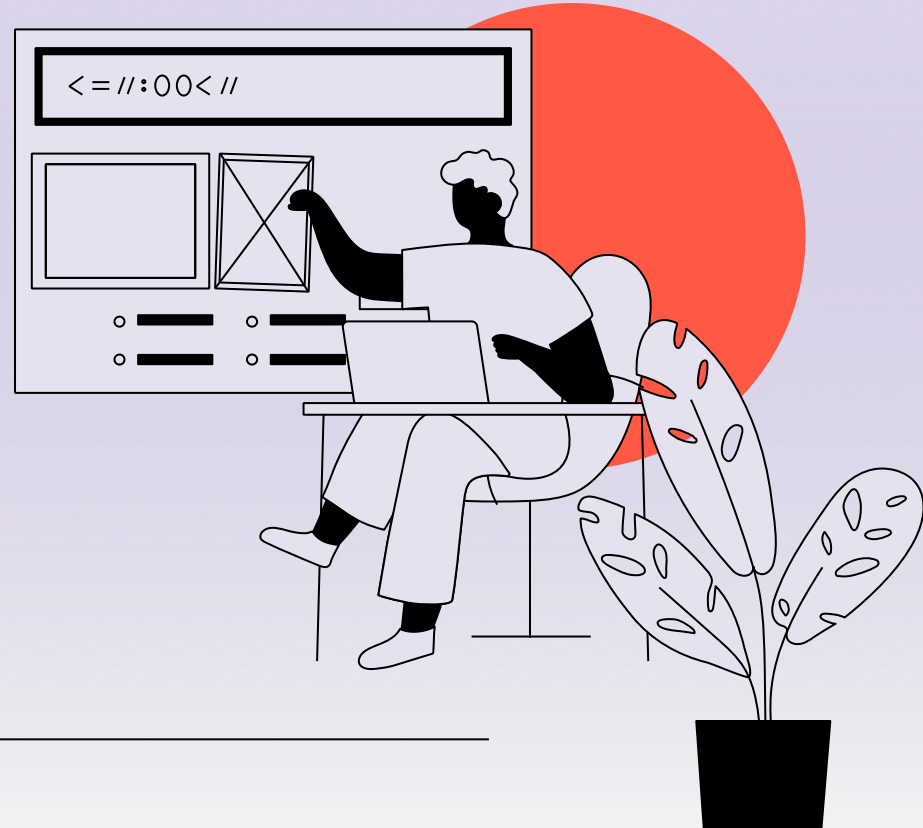
# Maintainability



- Adapt system to new cases.
- Design should be adaptable to change in the future.
- Future changes should be less costly.



An application has to meet various requirements in order to be useful. There are functional requirements (what it should do, such as allowing data to be stored, retrieved, searched) and some nonfunctional requirements like security, reliability, compliance, scalability, compatibility, and maintainability).



# The Theory of programming



Do you know what are the best practices for programming and why ?

- Values
- Principles
- Patterns

These three elements form a balanced expression of a style of development. The patterns describe what to do. The values provide motivation. The principles help translate motive into action.

# Values



## Communication

Think of others while program , easy to read then it's easy to modify it.



## Simplicity

good programs are written with an audience in mind.

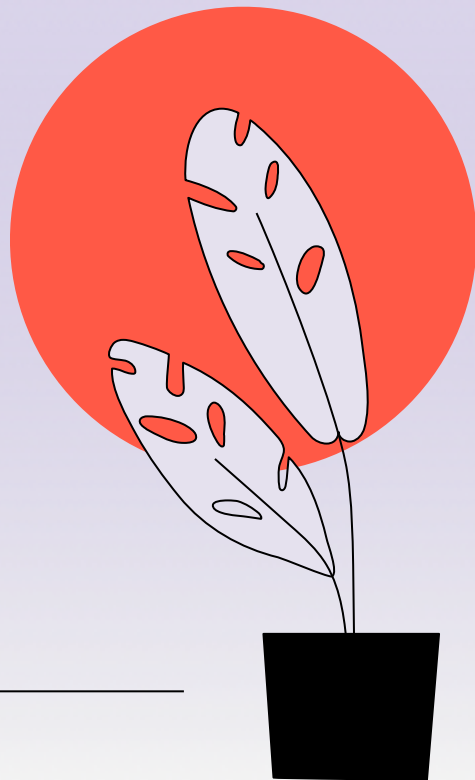


## Flexibility



# Principles

when I encounter a new programming language I use my understanding of principles to develop an effective style of programming.



# Local consequences

if a change here can cause a problem there, then the cost of the change rises dramatically, keep the cost of making changes low



---

# Minimize Repeatiation

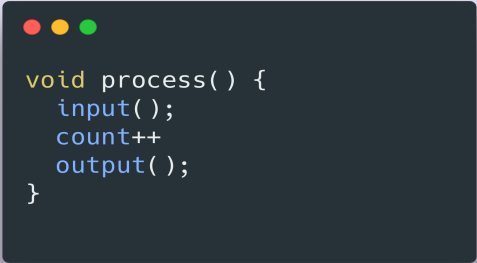
If making one conceptual change requires me to change two or more class hierarchies, then the changes have spreading consequences , One of the ways to remove duplication is to break programs up into many small pieces—small statements, small methods, small objects, small packages.



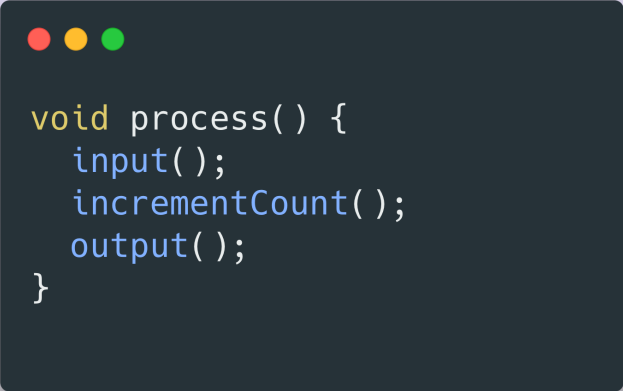
# Symmetry

Symmetry in code is where the same idea is expressed the same way everywhere it appears in the code.

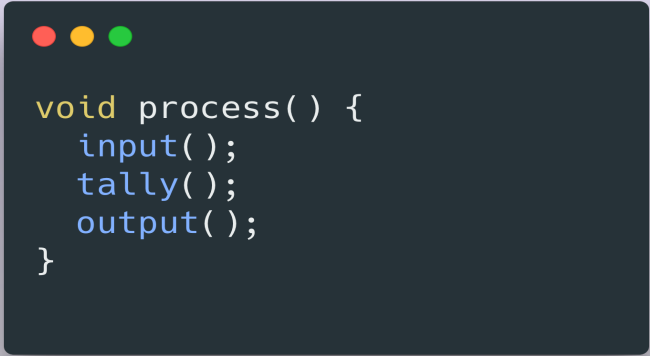




```
void process() {  
    input();  
    count++;  
    output();  
}
```



```
void process() {  
    input();  
    incrementCount();  
    output();  
}
```



```
void process() {  
    input();  
    tally();  
    output();  
}
```

---

---

# Rate Of Change

A final principle is to put logic or data that changes at the same rate together and separate logic or data that changes at different rates.





```
setAmount(int value, String currency) { this.value=
value;
this.currency= currency;
}
```



```
setAmount(int value, String currency) { this.value=
new Money(value, currency);
}
```



```
setAmount(Money value) { this.value= value;
}
```

---

One of the advantages of laying out a programming style as values, principles, and practices is that it is easier to have productive conflict about programming this way. If you want to do something one way and I another, we can identify the level of our disagreement and avoid wasting time. If we disagree about principles, arguing about where curly braces belong won't solve the underlying discord.

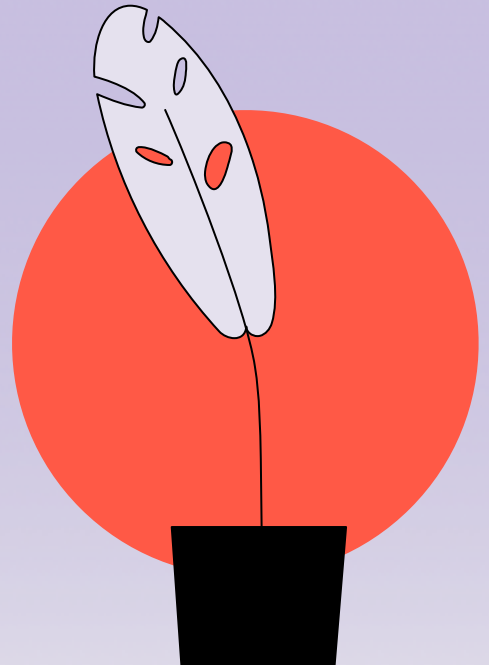
—Source: **Implementation Patterns**, Book by Kent Beck





---

# Code Smells





```
public void doSomething(String name, int id,  
String deptCode, String regNumber) {  
  
    ...  
}
```



```
public void doSomething(Student student) {  
    ...  
}
```




```
int id = obj.getDept().getSubDept().getHOD().getId();
```


---



```
public class Bird {  
  
    void fly() {  
        System.out.println("Flying!!");  
    }  
}  
  
public class Ostrich extends Bird {  
  
    void fly() {  
        throw new IllegalStateException("An ostrich can't  
fly");  
    }  
}
```



```
class Animal {  
    String type;  
  
    String makeSound() {  
        switch (type) {  
            case "cat":  
                return "meow";  
            case "dog":  
                return "woof";  
            default:  
                throw new IllegalStateException();  
        }  
    }  
}
```



```
abstract class Animal {  
    abstract String makeSound();  
}
```

```
class Cat extends Animal {  
    @Override  
    String makeSound() {  
        return "meow";  
    }  
}
```

```
class Dog extends Animal {  
    @Override  
    String makeSound() {  
        return "woof";  
    }  
}
```



```
var arr = [5, 7, 4, 12];  
var newArr = [];  
var i = 0;  
for(i=0;i<arr.length;i++)  
{  
    newArr.push(arr[i]*7)  
}  
console.log(newArr);
```



```
var arr = [5, 7, 4, 12];  
let map = arr.map(x => x * 7);  
console.log(map);
```





```
internal void AddCustomer(string customerName, string  
address, string phoneNumber) { }
```

```
internal void UpdateCustomer(string customerName,  
string address, string phoneNumber) { }
```

```
internal void SetCustomerEmail(string customerName,  
string email) { }
```

---

# Thanks

**Any questions?**



[marcel.adel55@gmail.com](mailto:marcel.adel55@gmail.com)



[linkedin.com/in/marcel-adel-b6681943](https://www.linkedin.com/in/marcel-adel-b6681943)

