1. DRY – Don't Repeat Yourself

Definition:
Avoid duplicating logic. Every piece of knowledge should exist **once and only once**.

Bad

```
if (user.getEmail() == null || user.getEmail().isEmpty()) { ... }
        if (admin.getEmail() == null || admin.getEmail().isEmpty()) { ... }
```

Good

```
boolean isValidEmail(String email) {
            return email != null && !email.isEmpty();
        }
```

2. KISS – Keep It Simple, Stupid

Definition:
Prefer the **simplest solution** that works. Avoid unnecessary complexity.

```
public boolean isEven(int number) {
            return number % 2 == 0 ? true : false;
        }
```

Why bad?

- Unnecessary ternary
- Verbose logic

Good

```
    public boolean isEven(int number) {
        return number % 2 == 0;
    }
```

Cleaner, clearer.

3. YAGNI – You Aren't Gonna Need It

Definition:
Do **not build features until they are actually required**.

Bad

```
class PaymentService {

    public void payByCreditCard() {
    }

    public void payByBitcoin() {
    }

    public void payByGoldCoins() {
    }
}
```

Problem:

Only credit card needed today
Extra code increases maintenance & bugs

```
class PaymentService {
    public void payByCreditCard() { }
}
```

When Bitcoin is needed → add it.

How These Work Together in Real Projects

Principle	Focus	Prevents
DRY	Duplication	Inconsistent logic / bugs
KISS	Complexity	Overengineering / confusion
YAGNI	Premature features	Wasted effort / bloated code

- **DRY** → "Write once, reuse everywhere"
- **KISS** → "Simple beats clever"
- **YAGNI** → "Build when needed, not imagined"