Behavior:

- John von Neumann introduces a primary metephors of computing
  → sequence of instructions
  → fundemental organizing principle

Control Flow:

- "Express computations as a sequence of steps."
- every program has a control of flow
  → except prolog

Main Flow:

- express the main flow of computation clearly is more valuable.
- focus on the main flow instead of little executed, seldom changed facts in the program
- main flow will be often read
  → exceptional conditional not

concentrate on invariant parts

Anforderung ist Anforderung
keine Anforderung

Message:

communicats something.

- fundamental control flow mechanism
- idea: change is the state change of programms.

→ methods: detail hidding mechanism, but not intention

clear and direct expressions of logic and defer details.
↳ skill, how to write programm, that communicates effectively

```
compute() {
  input();
  process();
  output();
}
```

- same level of abstraction
- message: three steps
- details are hiddin.
- use intention revealing names

Choosing Message:

→ to understand the reader have to look at several classes
↳ that's why use intention-revealing names

- this code send a polymorphic message.
  ↳ communicates, that a choice will take place at runtime.
- display choose the implementation at runtime.

```
public void displayShape(Shape subject, Brush brush) {
  brush.display(subject);
}
```

→ invitation for later expressions.
↳ wave of further innovations.

→ no variation of computation → choose message is overkill → YAGNI.

Double Dispatch :

- very the implementors of messages along two axes to express cascading choices.

choosing message → simple dimension of variability

```
public void displayShape(Shape subject, Brush brush) {
  brush.display(subject);
}
```

one dimension ⇒ on what medium display the subject

double dispatch → cascade two choosing messages
→ two independent dimensions of variability

```
displayShape(Shape subject, Brush brush) {
  shape.displayWith(brush);
}
```

```
Oval.displayWith(Brush brush) {
  brush.displayOval(this);
}
Rectangle.displayWith(Brush brush) {
  brush.displayRectangle(this);
}
```

```
PostscriptBrush.displayRectangle(Rectangle subject) {
  writer print(subject.left() +" " +...+ " rect);
}
```