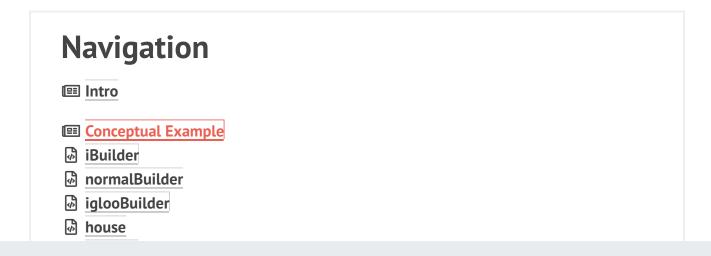


Builder is a creational design pattern, which allows constructing complex objects step by step.

Unlike other creational patterns, Builder doesn't require products to have a common interface. That makes it possible to produce different products using the same construction process.

■ Learn more about Builder →



Home Refactoring Design Patterns Premium Content Forum Contact us







The Builder pattern is used when the desired product is complex and requires multiple steps to complete. In this case, several construction methods would be simpler than a single monstrous constructor. The potential problem with the multistage building process is that a partially built and unstable product may be exposed to the client. The Builder pattern keeps the product private until it's fully built.

In the below code, we see different types of houses (igloo and normalHouse) being constructed by iglooBuilder and normalBuilder. Each house type has the same construction steps. The optional director struct helps to organize the building process.

iBuilder.go: Builder interface

```
type IBuilder interface {
    setWindowType()
    setDoorType()
    setNumFloor()
    getHouse() House
}

func getBuilder(builderType string) IBuilder {
    if builderType == "normal" {
        return newNormalBuilder()
    }

    if builderType == "igloo" {
        return newIglooBuilder()
    }

    return nil
}
```

🖟 normalBuilder.go: Concrete builder

```
package main

type NormalBuilder struct {
   windowType string
   doorType string
   floor int
```

```
func newNormalBuilder() *NormalBuilder {
   return &NormalBuilder{}
}
func (b *NormalBuilder) setWindowType() {
   b.windowType = "Wooden Window"
}
func (b *NormalBuilder) setDoorType() {
    b.doorType = "Wooden Door"
}
func (b *NormalBuilder) setNumFloor() {
   b.floor = 2
}
func (b *NormalBuilder) getHouse() House {
   return House{
        doorType:
                    b.doorType,
        windowType: b.windowType,
        floor:
                    b.floor,
   }
}
```

🖟 iglooBuilder.go: Concrete builder

```
type IglooBuilder struct {
    windowType string
    doorType string
    floor int
}

func newIglooBuilder() *IglooBuilder {
    return &IglooBuilder{}
}

func (b *IglooBuilder) setWindowType() {
    b.windowType = "Snow Window"
}
```

```
func (b *IglooBuilder) setDoorType() {
    b.doorType = "Snow Door"
}

func (b *IglooBuilder) setNumFloor() {
    b.floor = 1
}

func (b *IglooBuilder) getHouse() House {
    return House{
        doorType: b.doorType,
            windowType: b.windowType,
            floor: b.floor,
        }
}
```

house.go: Product

```
type House struct {
    windowType string
    doorType string
    floor int
}
```

director.go: Director

```
package main

type Director struct {
    builder IBuilder
}

func newDirector(b IBuilder) *Director {
    return &Director{
        builder: b,
     }
}

func (d *Director) setBuilder(b IBuilder) {
```

```
d.builder = b
}

func (d *Director) buildHouse() House {
    d.builder.setDoorType()
    d.builder.setWindowType()
    d.builder.setNumFloor()
    return d.builder.getHouse()
}
```

🖟 main.go: Client code

```
import "fmt"

func main() {
    normalBuilder := getBuilder("normal")
    iglooBuilder := getBuilder("igloo")

    director := newDirector(normalBuilder)
    normalHouse := director.buildHouse()

fmt.Printf("Normal House Door Type: %s\n", normalHouse.doorType)
    fmt.Printf("Normal House Window Type: %s\n", normalHouse.windowType)
    fmt.Printf("Normal House Num Floor: %d\n", normalHouse.floor)

director.setBuilder(iglooBuilder)
    iglooHouse := director.buildHouse()

fmt.Printf("\nTgloo House Door Type: %s\n", iglooHouse.doorType)
    fmt.Printf("Tgloo House Window Type: %s\n", iglooHouse.windowType)
    fmt.Printf("Tgloo House Window Type: %s\n", iglooHouse.windowType)
    fmt.Printf("Tgloo House Num Floor: %d\n", iglooHouse.floor)
}
```

output.txt: Execution result

```
Normal House Door Type: Wooden Door
Normal House Window Type: Wooden Window
Normal House Num Floor: 2
```

Igloo House Door Type: Snow Door Igloo House Window Type: Snow Window

Igloo House Num Floor: 1

Based on: Golang By Example

RETURN READ NEXT

← Abstract Factory in Go

Factory Method in Go →

Builder in Other Languages

















