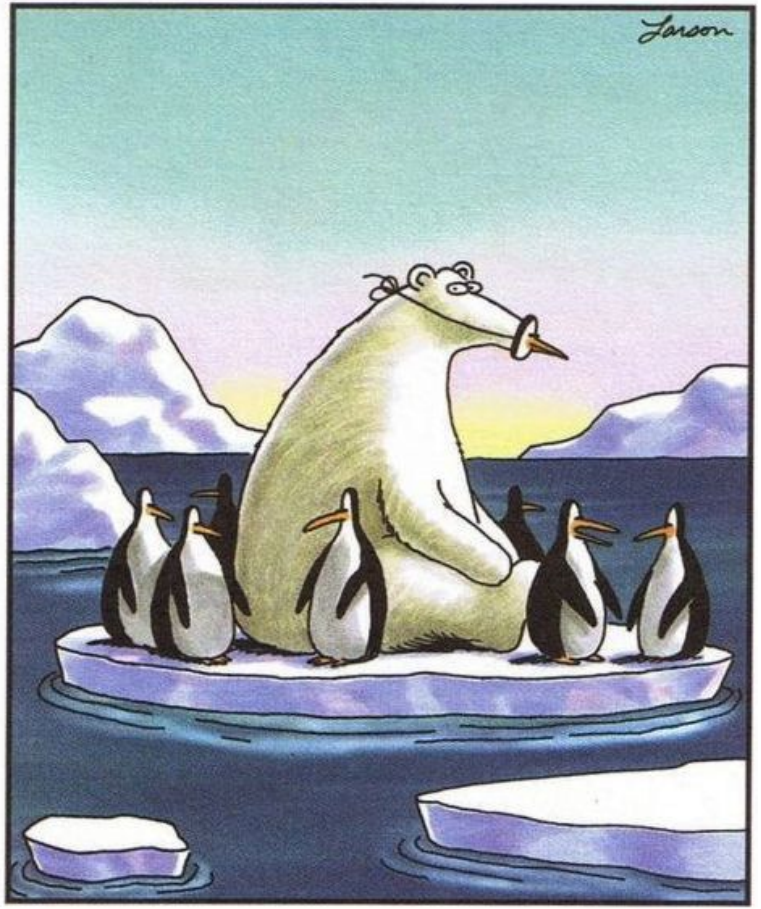


And now Edgar's gone. ...
Something's going on around here.

Hoja de Ruta

Semana 15 Abril

- Patrones Construcción 1 (Factory Method)

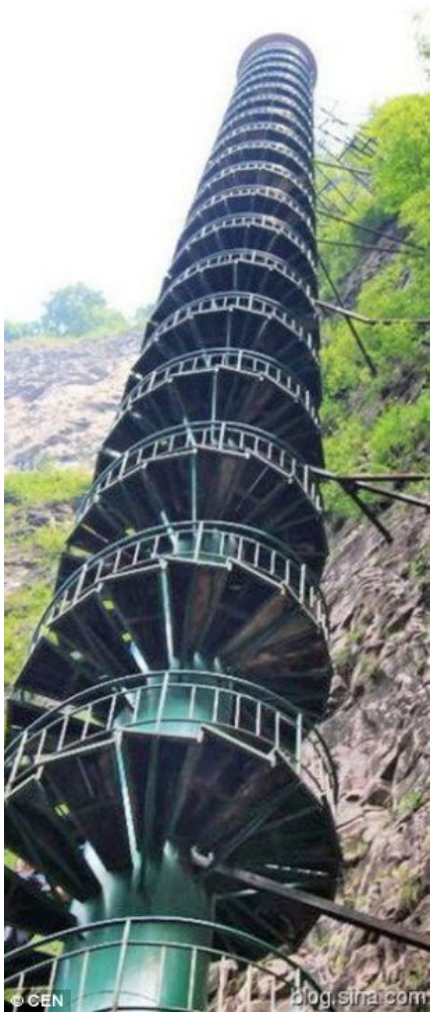
Semana 23 Abril

- **Patrones de Dominio 1 (Null Object)**
- **Patrones Construcción 2 (Builder)**

Semana 30 Abril

- Patrones de Dominio 2 (TypeObject)
- Lenguaje de Patrones: Test Doubles

Federico Balaguer: federico.balaguer@lifia.unlp.edu.ar



¿Quién ha visto este tipo de código?

```
Customer customer = findCustomerById(10);

if (customer != null) {
  System.out.println("Customer email: " + customer.getEmail());
} else {
  System.out.println("Customer with ID 10 not found.");
}
```

```
27  Customer customer = findCustomerById(10);
28
29  if (customer != null) {
30    String email = customer.getEmail();
31    if (email != null){
32      System.out.println("Customer email: " + email);
33    } else{System.out.println("Customer email: not in the system");
34    }
35  } else {
36    System.out.println("Customer with ID 10 not found.");
37  }
```

¿Que significa?

¿Por qué tantos condicionales? ¿Cual es el problema?

```
40   Customer customer = findCustomerById(10);
41
42   if (customer != null) {
43     Riesgo riesgo = customer.getRiesgo();
44     if (riesgo != null){
45       riesgo.actualizarRangoFechasHoy();
46     } else{//no hacemos nada}
47     }
48   } else {//nada por hacer
49     System.out.println("Customer with ID 10 not found.");
50   }
```

Null Object Pattern (Woolf&Johnson)

Objetivo: Proporciona un sustituto para otro objeto que comparte la misma interfaz pero no hace nada. El NullObject encapsula las decisiones de implementación de cómo "no hacer nada" y oculta esos detalles de sus colaboradores

Alias: Stub

Consecuencia:

- Elimina todos los condicionales que verifican si la referencia a un objeto es NULL
- Hace explícito elementos del dominio que hacen “nada”

Ahora que saben de que se trata lean el paper :-)

Así quedaría el código gracias a NullObject para Customer y Riesgo

```
Customer customer = findCustomerById(10);
Riesgo riesgo = customer.getRiesgo();
riesgo.actualizarRangoFechasHoy();
```

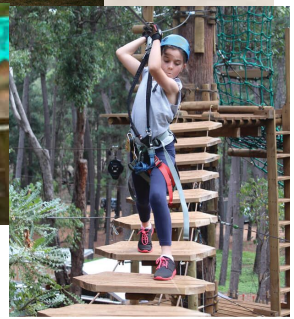
```
Customer customer = findCustomerById(10);
(customer.getRiesgo()).actualizarRangoFechasHoy();
```

```
((this.findCustomerById(10)) customer.getRiesgo()).actualizarRangoFechasHoy();
```

- Transporte
- Alojamiento
- Asistencia Médica
- Coordinación (profes y Guardavidas)
- Fiesta
- Excursiones

El esquema general es el mismo

- Transporte
- Alojamiento c/comidas
- Asistencia Médica
- Coordinación
  - Profes
  - Guardavidas
- Fiesta
- Excursiones

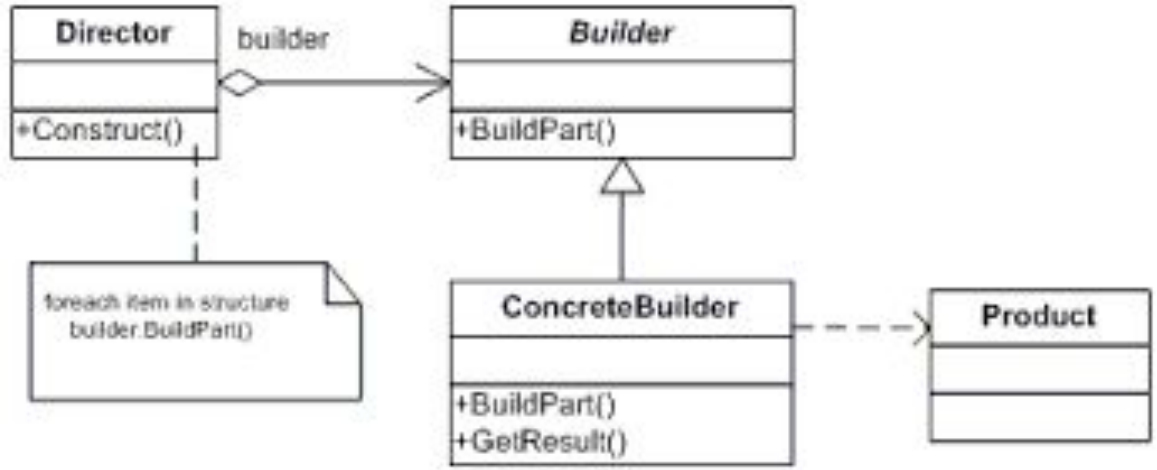


Cual es el problema?

- Cambios de estado de un objeto en tiempo de ejecución?
- Una definición única que puede ser “customizable” a casos específicos?
- Adaptar protocolos entre objetos?
- Manejar la configuración de objetos para que tengan diferente comportamiento?
  - Pero existe una definición reusable que se aplica a todos los casos
  - La "estructura" se repite en la estructura del producto pero las partes pueden ser diferentes en cada caso.

Builder

- Intención: separa la construcción de un objeto complejo de su representación (implementación) de tal manera que el mismo proceso puede construir diferente representaciones (implementaciones)
- Estructura (**Roles**)

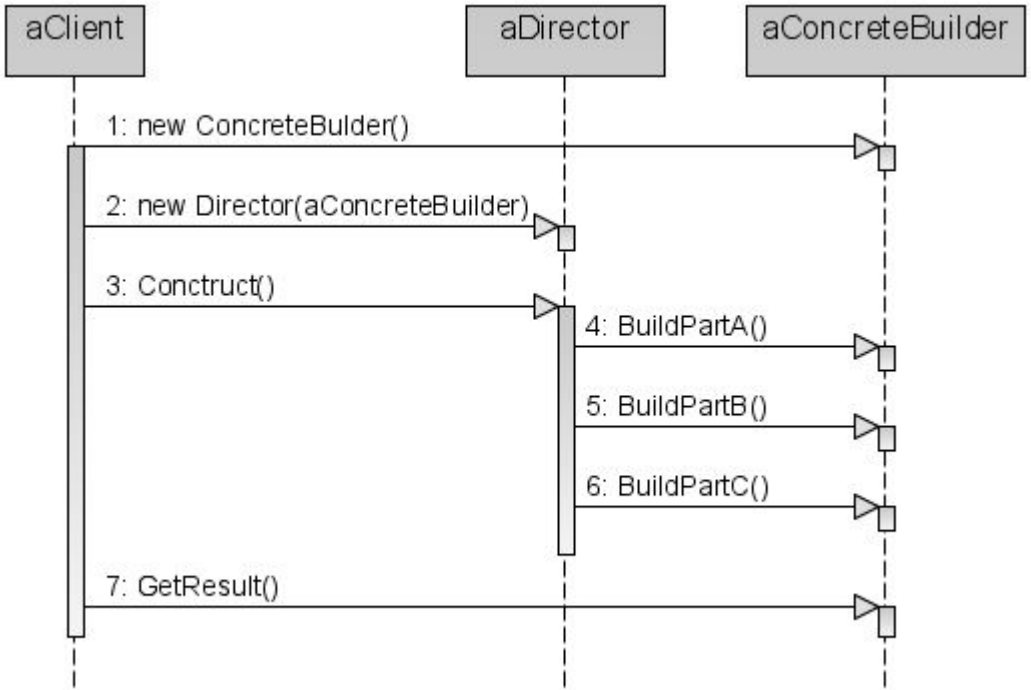


Builder

Participantes

- Builder: especifica una interface abstracta para crear partes de un Producto
- Concrete Builder: construye y ensambla partes del producto.
  - Guarda referencia al producto en construccion
- Director: conoce los pasos para construir el objeto
  - Utiliza el Builder para construir las partes que va ensamblando
  - En lugar de pasos fijos puede seguir una “especificación” (ver knwon uses@ GOF)
- Product: es el objeto complejo a ser construido

Diagrama de Secuencia



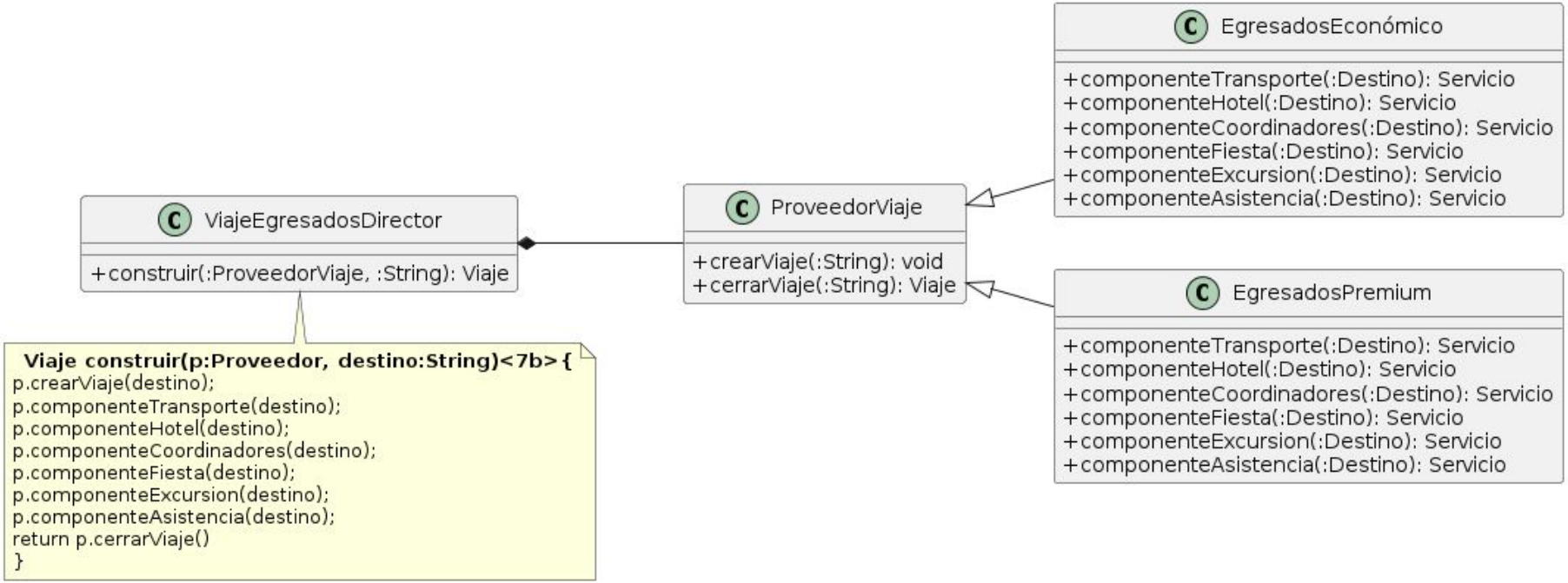
Vale la pena Builder?

Si!

- Abstrae la construcción compleja de un objeto complejo
- Permite variar lo que se construye Director <-> Builder
- Da control sobre los pasos de contrucción

No!

- Requiere diseñar y implementar varios roles
- Cada tipo de producto requiere un ConcreteBuilder
- Builder suelen cambiar o son parsers de specs (> complejidad)


EgresadosEconómico
+componenteTransporte(:Destino): Servicio
+componenteHotel(:Destino): Servicio
+componenteCoordinadores(:Destino): Servicio
+componenteFiesta(:Destino): Servicio
+componenteExcursion(:Destino): Servicio
+componenteAsistencia(:Destino): Servicio
ViajeEgresadosDirector
+construir(:ProveedorViaje, :String): Viaje
ProveedorViaje
+crearViaje(:String): void
+cerrarViaje(:String): Viaje
EgresadosPremium
+componenteTransporte(:Destino): Servicio
+componenteHotel(:Destino): Servicio
+componenteCoordinadores(:Destino): Servicio
+componenteFiesta(:Destino): Servicio
+componenteExcursion(:Destino): Servicio
+componenteAsistencia(:Destino): Servicio
Viaje construir(p:Proveedor, destino:String)<7b>{
p.crearViaje(destino);
p.componenteTransporte(destino);
p.componenteHotel(destino);
p.componenteCoordinadores(destino);
p.componenteFiesta(destino);
p.componenteExcursion(destino);
p.componenteAsistencia(destino);
return p.cerrarViaje()
}

Consideraciones sobre Builder

1. El Director **solo** sabe hacer **una** cosa
2. Los Builders pueden saber hacer cosas que no requiera un Director pero si otro
   a. Ej: componentePsicologico()
3. Otros Directors pueden usar los mismos Builders
4. Nuevas definiciones de Viajes de Egresado ⇒nuevos directores
5. Nuevos servicios ⇒ nuevos Builders

Preguntas de Final…

1)¿Cómo se implementa un viaje a Punta del Este?

2) ¿Cómo se implementa un viaje a Río de Janeiro con aéreo y traductor?

3) ¿Cómo se implementa un viaje con múltiples destinos? Ej: Tandil y Miramar

Para la semana que viene

- Leer Type-Object
  - Problema de las piezas de ajedrez: mismo movimiento diferente color
  - Seguro de autos: Servicios visto como Coberturas o Reclamos
- Completen la siguiente tabla

	FactoryMethod	**Builder**
Problema	Construcción	Construcción
Encapsula complejidad al cliente?	si/no	si/no
Definición unica ? (receta)	si/no	si/no
Identifica proveedor de partes?	si/no	si/no
Simil constructor?	si/no	si/no
Primer estadío de re-diseño? (usualmente)	si/no	si/no