

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 22-b: Solución Mario Maker (Builder)

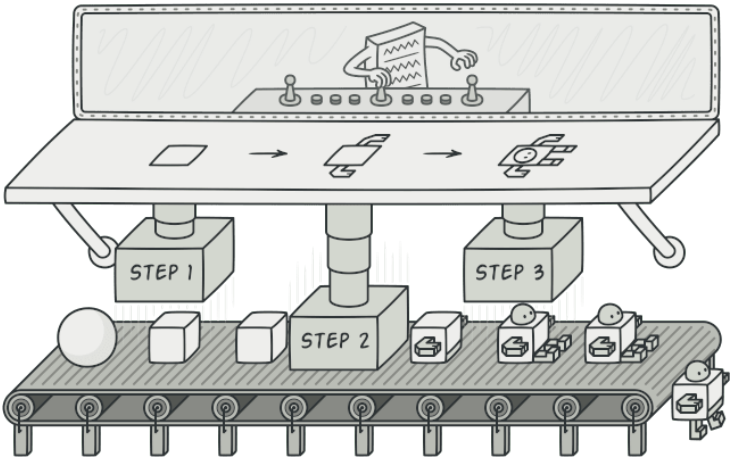
Francisco Gazitúa Requena (fco_gazitua_requena@uc.cl)

IIC2113 Diseño Detallado de Software

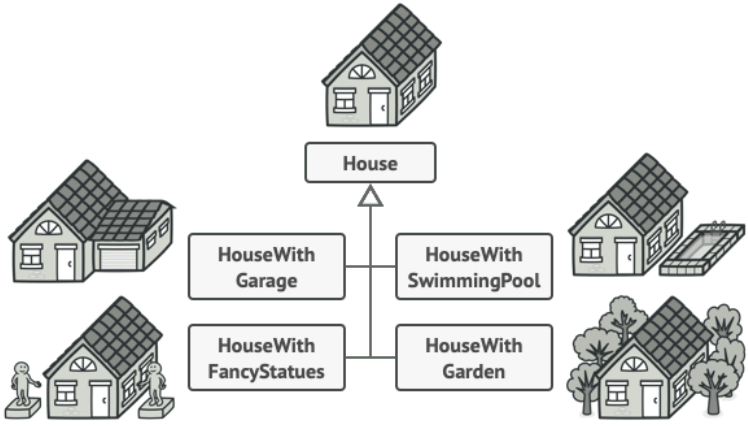
23 de Octubre, 2025

Mario Maker (Builder)

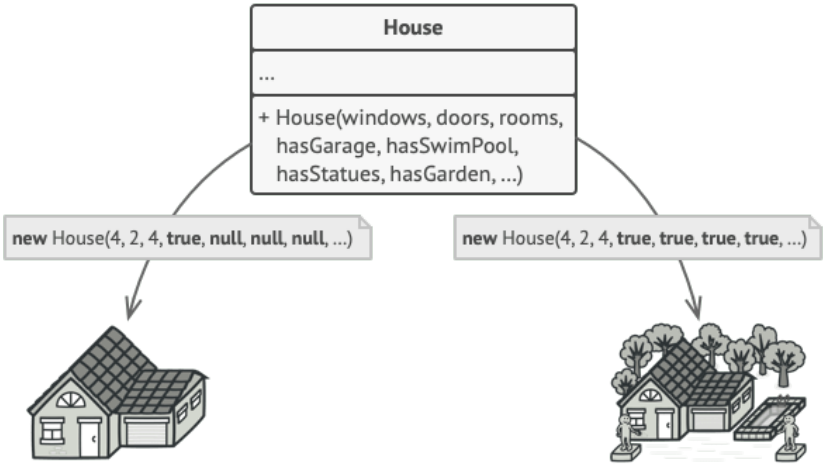
Mario Maker (Builder)



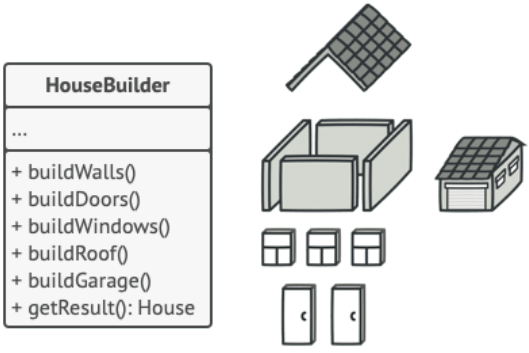
Mario Maker (Builder)



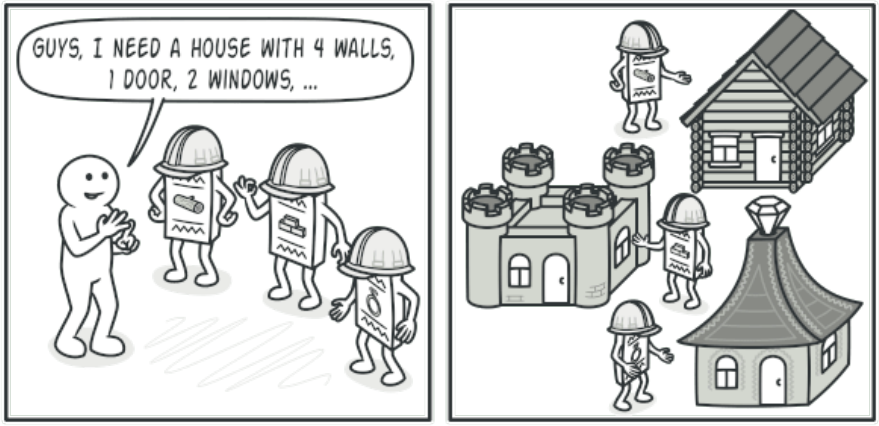
Mario Maker (Builder)



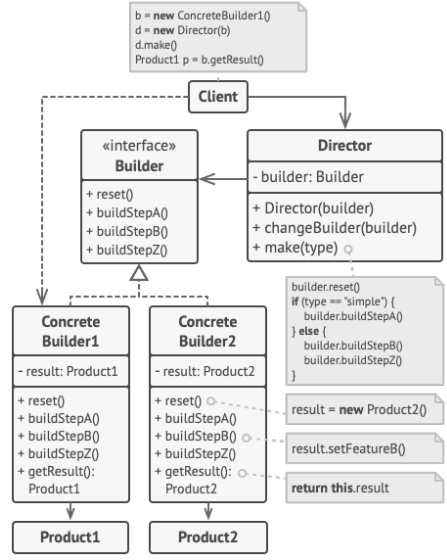
Mario Maker (Builder)



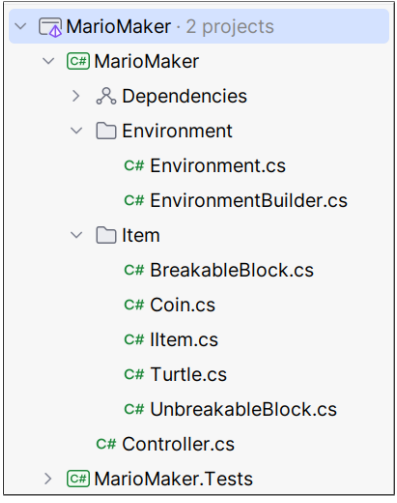
Mario Maker (Builder)



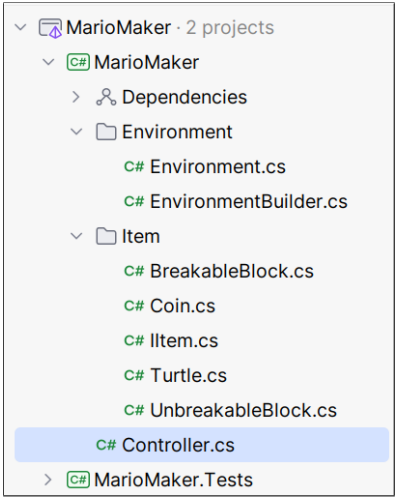
Mario Maker (Builder)



Mario Maker (Builder)



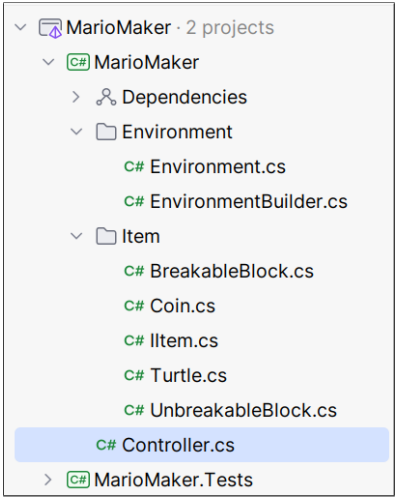
Mario Maker (Builder)



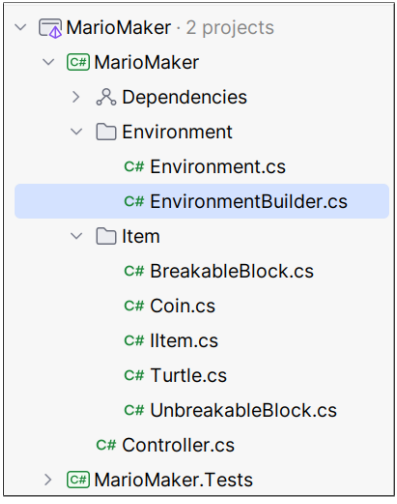
Mario Maker (Builder)

```
1  public static class Controller {
2
3    public static Environment BuildEnvironment(EnvironmentBuilder builder,
       string[] lines) {
4      throw new NotImplementedException();
5    }
6
7    public static EnvironmentBuilder BuildCloudEnvironmentBuilder() {
8      throw new NotImplementedException();
9    }
10
11   public static EnvironmentBuilder BuildLavaEnvironmentBuilder() {
12     throw new NotImplementedException();
13   }
14
15   public static string[] Export(Environment environment) {
16     throw new NotImplementedException();
17   }
18 }
```

Mario Maker (Builder)



Mario Maker (Builder)



Mario Maker (Builder)

```
public class EnvironmentBuilder {
  private Environment _environment;

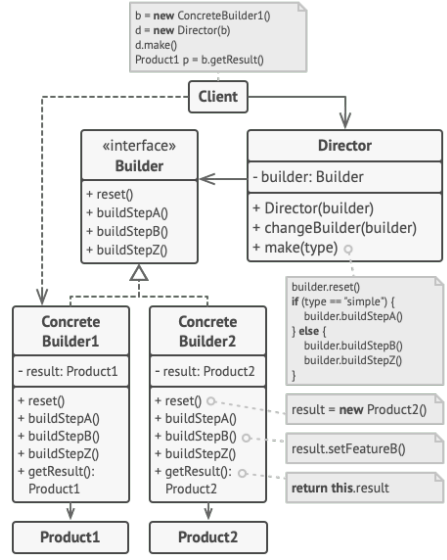
  public void Reset()
    => _environment = new Environment();

  public void SetDimensions(int height, int width)
    => _environment.SetDimensions(height, width);

  public void SetUnbreakableBlock(int row, int column)
    => _environment.SetItem(row, column, new UnbreakableBlock());
  public void SetBreakableBlock(int row, int column)
    => _environment.SetItem(row, column, new BreakableBlock());
  public void SetTurtle(int row, int column)
    => _environment.SetItem(row, column, new Turtle());
  public void SetCoin(int row, int column)
    => _environment.SetItem(row, column, new Coin());

  public Environment GetResult()
    => _environment;
}
```

Mario Maker (Builder)



Mario Maker (Builder)

Parte a) Actualmente hay un programa para generar *templates* de niveles, los cuales son archivos de texto que representan una grilla. El número de líneas dentro del archivo representa la altura de la grilla y el largo de las líneas el ancho (puedes asumir que todas las líneas del archivo tienen el mismo largo).

Cada caracter (en caso de haber alguno) representa una entidad del ambiente:

- U: `UnbreakableBlock`.
- B: `BreakableBlock`.
- C: `Coin`.
- T: `Turtle`.

Implementa el método `BuildEnvironment(EnvironmentBuilder builder, string[] lines)` dentro de `Controller.cs`. Este método debe retornar un objeto del tipo `Environment` que sea construido a partir de un *template*, el cual está dado por el argumento `lines`.

Mario Maker (Builder)

```
1  public static class Controller {
2
3    public static Environment BuildEnvironment(EnvironmentBuilder builder,
       string[] lines) {
4      throw new NotImplementedException();
5    }
6
7    public static EnvironmentBuilder BuildCloudEnvironmentBuilder() {
8      throw new NotImplementedException();
9    }
10
11   public static EnvironmentBuilder BuildLavaEnvironmentBuilder() {
12     throw new NotImplementedException();
13   }
14
15   public static string[] Export(Environment environment) {
16     throw new NotImplementedException();
17   }
18 }
```

Mario Maker (Builder)

```
public static Environment BuildEnvironment(EnvironmentBuilder builder, string
    [] lines) {
  builder.Reset();

  int height = lines.Length;
  int width = lines[0].Length;
  builder.SetDimensions(height, width);
  for (int i = 0; i < height; i++)
    for (int j = 0; j < width; j++)
    {
      if(lines[i][j] == 'U')
        builder.SetUnbreakableBlock(i, j);
      if(lines[i][j] == 'T')
        builder.SetTurtle(i, j);
      if(lines[i][j] == 'C')
        builder.SetCoin(i, j);
      if(lines[i][j] == 'B')
        builder.SetBreakableBlock(i, j);
    }

  return builder.GetResult();
}
```

Mario Maker (Builder)

Parte b) Extiende el patrón `Builder` para generar niveles ambientados en el mundo de lava y en el mundo de nubes. Para ello implementa:

- `BuildLavaEnvironmentBuilder()`
- `BuildCloudEnvironmentBuilder()`

Por ahora, la única diferencia entre estos mundos es el tipo de tortugas que se encuentran en ellos. En el mundo de lava solo existen tortugas de lava y en el mundo de nube solo existen tortugas con alas.

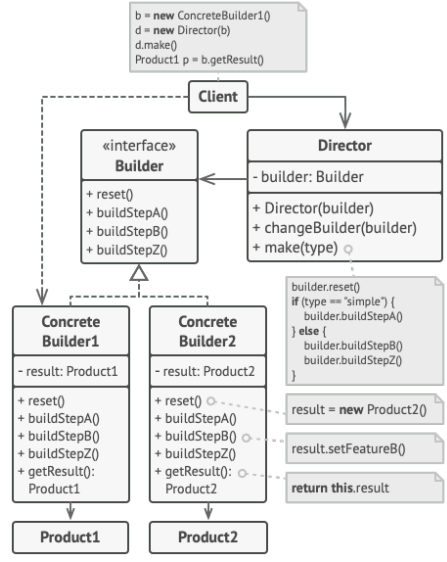
En específico, el método `Export()` de las tortugas del mundo de lava debe retornar "LavaTurtle" y las del mundo de nube debe retornar "WingedTurtle".

Mario Maker (Builder)

Nos piden retornar builders para estos nuevos tipos de ambientes:

```
1  public static class Controller {
2
3    public static Environment BuildEnvironment(EnvironmentBuilder builder,
       string[] lines) {
4      /* ... idem anterior ... */
5    }
6
7    public static EnvironmentBuilder BuildCloudEnvironmentBuilder(){
8      throw new NotImplementedException();
9    }
10
11   public static EnvironmentBuilder BuildLavaEnvironmentBuilder(){
12     throw new NotImplementedException();
13   }
14
15   public static string[] Export(Environment environment) {
16     throw new NotImplementedException();
17   }
18 }
```

Mario Maker (Builder)



Mario Maker (Builder)

```
public class EnvironmentBuilder {
  private Environment _environment;

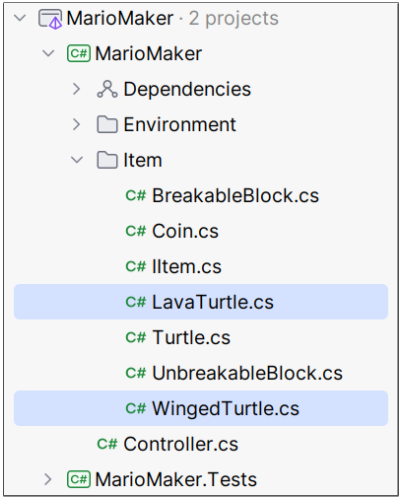
  public void Reset()
    => _environment = new Environment();

  public void SetDimensions(int height, int width)
    => _environment.SetDimensions(height, width);

  public void SetUnbreakableBlock(int row, int column)
    => _environment.SetItem(row, column, new UnbreakableBlock());
  public void SetBreakableBlock(int row, int column)
    => _environment.SetItem(row, column, new BreakableBlock());
  public void SetTurtle(int row, int column)
    => _environment.SetItem(row, column, new Turtle());
  public void SetCoin(int row, int column)
    => _environment.SetItem(row, column, new Coin());

  public Environment GetResult()
    => _environment;
}
```

Mario Maker (Builder)



Mario Maker (Builder)

Esta es la tortuga de lava:

```
6  public class LavaTurtle : Turtle {
7    public override string Export()
8      => "LavaTurtle";
9  }
```

Esta es la tortuga con alas:

```
11 public class WingedTurtle : Turtle{
12   public override string Export()
13     => "WingedTurtle";
14 }
```

Mario Maker (Builder)

Esta es la tortuga de lava:

```
6  public class LavaTurtle : Turtle {
7    public override string Export()
8      => "LavaTurtle";
9  }
```

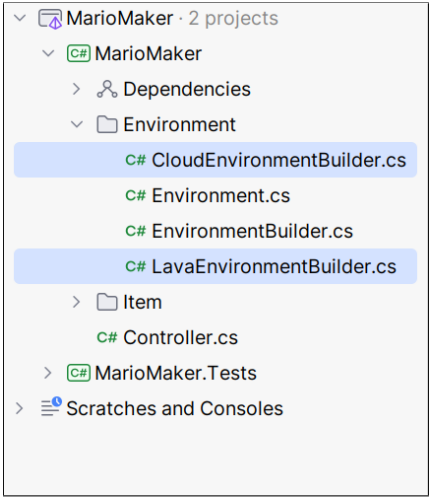
Esta es la tortuga con alas:

```
11 public class WingedTurtle : Turtle{
12   public override string Export()
13     => "WingedTurtle";
14 }
```

... para que esto funcione tenemos que hacer que `Export()` `virtual` en `Turtle`:

```
1  public class Turtle : IItem {
2    public virtual string Export() // Ahora es virtual!
3      => "Turtle";
4  }
```

Mario Maker (Builder)



Mario Maker (Builder)

Builder de lava (todo igual menos la tortuga):

```
1 public class LavaEnvironmentBuilder : EnvironmentBuilder {
2   public override void SetTurtle(int row, int column)
3     => Environment.SetItem(row, column, new LavaTurtle());
4 }
```

Builder de nube (todo igual menos la tortuga):

```
6 public class CloudEnvironmentBuilder: EnvironmentBuilder {
7   public override void SetTurtle(int row, int column)
8     => Environment.SetItem(row, column, new WingedTurtle());
9 }
```

Mario Maker (Builder)

Builder de lava (todo igual menos la tortuga):

```
1  public class LavaEnvironmentBuilder : EnvironmentBuilder {
2    public override void SetTurtle(int row, int column)
3      => Environment.SetItem(row, column, new LavaTurtle());
4  }
```

Builder de nube (todo igual menos la tortuga):

```
6  public class CloudEnvironmentBuilder: EnvironmentBuilder {
7    public override void SetTurtle(int row, int column)
8      => Environment.SetItem(row, column, new WingedTurtle());
9  }
```

... para que esto funcione tenemos que hacer cambios menores al builder base:

```
11 public class EnvironmentBuilder{
12   protected Environment Environment; // Pasa a protected
13   public virtual void SetTurtle(int row, int column) // Pasa a virtual
14     => Environment.SetItem(row, column, new Turtle());
15   /* ... el resto queda igual... */
16 }
```

Mario Maker (Builder)

Ahora simplemente basta con retornar el builder apropiado en el `Controller`:

```
1  public static class Controller {
2
3    public static Environment BuildEnvironment(EnvironmentBuilder builder,
       string[] lines) {
4      /* ... idem anterior ... */
5    }
6
7    public static EnvironmentBuilder BuildCloudEnvironmentBuilder()
8      => new CloudEnvironmentBuilder();
9
10   public static EnvironmentBuilder BuildLavaEnvironmentBuilder()
11     => new LavaEnvironmentBuilder();
12
13   public static string[] Export(Environment environment) {
14     throw new NotImplementedException();
15   }
16 }
```