|  |  |
| --- | --- |
| **Nombre** | **Patrón Builder** |
| **Propósito** | Permite crear un todo o las partes de un todo sin la necesidad de crear siempre todo el objeto. |
| **Problema** | Generar campamentos con sus partes o espacios. |
| **Solución** | Se crea el código utilizando el patrón builder. |
| **Participantes y Colaboradores** | |
| constructores | Builder, ConstruirAcademiaPiloto, ConstruirCampoEntrenamiento, ConstruirEscuelaInfanteria, ConstruirEscuelaArtilleria, CosntruirEscuelaMedicina, CosntruirTorreVigilancia, ConstruirMuralla, ConstruirCampamento. |
| directores | Director. |
| main | Gestor, Main. |
| objetos | Edificacion. |
| **Consecuencias** | |
| Se pueden variar libremente la implementación interna de los objetos. | |
| Se aisla la construcción de la representación de los objetos. | |
| Se controla completamente el proceso de construcción de los  mismos. | |
| **Implementación** | |
| El cliente puede construir un campamento y sus partes: Campo de entrenamiento, escuela de infantería, escuela de artillería, Academia de pilotos, Escuela de medicina, Torre de vigilancia, Muralla, de acuerdo a lo que requiera gracias a la clase abstracta Builder. | |
| Gracias a esto se puede crear todo el campamento o en el caso de que ya tenga el campamento y le haga falta una parte o quera tener una extra puede crear esa parte sin necesidad de volver a crear otro campamento entero. | |
| Así el sistema es independiente y no necesita crear todos los objetos de nuevo. Al cliente no le preocupa como se crean los objetos solo los pide y listo. | |

UML

