Design Patterns

Here are the Design Patterns we have learned about and implemented in our project.
Sources we have used:

1. https://en.wikipedia.org/wiki/Factory_method_pattern
2. https://refactoring.guru/design-patterns
3. https://www.baeldung.com/java-factory-pattern
4. https://ronnieschaniel.medium.com/object-oriented-design-patterns-explained-using-practical-examples-84807445b092

Name	Description/Benefits	In our code
Factory Method	Makes generating similar objects easier. You can use a Factory (in this case, EntityFactory) without knowing what exact kind of object (Entity) it creates. Factories can create objects of different classes, but also depend on some common initialization code. Our adaptation: We have created methods to allow an EntityFactory to create objects on a schedule and add them to a list.	Used in Game, BlockGrid Implemented in EntityFactory, BlockFactory, EnemyFactory
State	Separate state-dependent behavior into separate classes. Remove long and hard-to-read if or switch statements. Lets several states share code (through inheritance).	Used in Block Implemented in BlockState, BlockStateStacked, BlockStateFalling etc.
Composite Object	Several objects can be interacted with through a singular object.	Implemented in BlockGrid In our case a BlockGrid is a single object implementing Entity, but calling Entity methods on it actually triggers them on many Block objects (which are also Entities). A Composite Object most often represents a tree-like hierarchy,

		however, in our case it is only one level deep, as Blocks themselves are not Composite Objects.
Facade	“A Facade simplifies the interface to an object or a group of objects ‘behind’ this facade.”	Implemented in BlockGrid, Game BlockGrid provides methods like addObject, removeObject, putBlockAt, getBlockAt, getBlocks etc. to interact with its internal list of Blocks and 2D array containing the Block grid. Game provides methods addEntity, removeEntity, getEntities to interact with its internal list of Entities, while also imposing some rules (for example, the player Entity cannot be removed).