Prototype design pattern:

Used when we have an instance of the class (prototype) and we’d like to create the **new object by just copying the prototype.**

In some games, we want trees or buildings in the background. We don’t want to create new trees or buildings to render them on the screen every time character moves.

Diagram

Description automatically generated

There are two types of cloning they are deep cloning and shallow cloning.

We’d implement the Prototype pattern using clone () method of cloneable interface.

If the class contains only **primitive** and **immutable** fields, we may use **shallow copy**.

If it **contains references to mutable fields**, we should be got for **deep copy**. We might do that with copy constructor or serialization or deserialization.

Prototype pattern also lets us create copies of objects without depending on concrete classes.

Advantage of using prototype design pattern

* We can add or remove the implementation while running.
* Reduced subclassing - Prototype pattern lets you clone a prototype instead of asking factory method to make a new object.
* Specifying new object by carrying structure.
* Specifying new objects by varying values

Disadvantage:

* Overkill for project that uses few objects and does not have underlying emphasis on extension of prototype chains.
* Hides concrete product classes from client.
* Implementation of clone can be difficult when their internals include objects that doesn’t support copying or have circular references.