Use the ***Prototype Pattern*** when creating an instance of a given class is either expensive or complicated.

***Scenario – Interactive role-playing game with monsters.***

The ***Prototype Pattern*** allows you to make new instances by copying existing instances. (In Java this typically means using the clone() method, or de-serialization when you need deep copies.) A key aspect of this pattern is that the client code can make new instances without knowing which specific class is being instantiated.

<<interface>>

Monster

DynamicPlayerGeneratedMonster

WellKnownMaster

MonsterRegistry

Monster getMonster() {

// find the correct monster

Return correctMonster.clone();

}

MonsterMaker

makeRandomMonster() {

Monster m = MonsterRegistry.getMonster();

}

***Benefits:***

* Hides the complexities of making new instances from the client.
* Provides the option for the client to generate objects whose type is not known.
* In some circumstances, copying an object can be more efficient than creating a new object.

***Uses and Drawbacks:***

* Prototype should be considered when a system must create new objects of many types in a complex class hierarchy.
* A drawback to using the Prototype is that making a copy of an object can sometimes be complicated.