Abstract Factory: Each factory in this example (OwlEmporium, MagicWandFactory, etc.) implements the Factory interface. This interface provides functionality for item delivery to the user with the definition of the abstract method, deliverItem(). This provides a vehicle for object creation without directly accessing the constructor for the objects associated with each factory (Owl, MagicWand, etc.)

Builder: There is a Potion class that is able to hold the name of the potion and a List of the ingredients. There are also different prebuilt potion making classes (DraughtOfPeacePotionMaker and WitSharpeningPotionMaker) which extend the abstract class PotionMaker. This abstract class defines the possible ingredients and defines an abstract method for making the potions, as well as methods for adding each available ingredient. The Director class chooses random combinations of ingredients and tries to make certain potions with them, switching out combinations of ingredients until the correct combination of ingredients is found.

Factory Method: There is a class called MagicWandMaker that generates a certain type of MagicWand object based on the skill level taken from the user through outside input. This is due to the makeArtifact() method in the MagicWandMaker class, which specifies which skill level is being used, being called in the main method. If the artifact is unable to be created, the ArtifactCannotBeCreated exception is thrown.

Mediator: There is an abstract class called Mediator that specifies the format in which the charges, judge, and witnesses are stored, as well as providing functionality to change each of these things. This class is extended by the MinistryOfMagicTrialMediator class, which gets each “state” of the trial stored in the HashMap specified by the Mediator abstract class. This is how the “trial” is able to advance, similar to how a trial in real life has different parts/stages to it.

Memento: The HogwartsHappening class has a method called saveStateToMemento() which returns a new Memento object (which is contained in the HogwartsHappening file.) This object can store the current storyline, as well as the time of the storyline’s occurrence. There is also a method called restoreStateFromMemento() that restores a Memento object specified in the method’s parameters.

Prototype: The Dragon class implements the Cloneable interface in order to be able to create different types of Dragons more easily. This class stores Set of the different possible types of Dragons available for creation. If the Dragon type specified as a parameter in the constructor of the Dragon class doesn’t match any of the Strings contained in the allTypes Set, the UnknownDragonType Exception is thrown. The Dragon class also contains a clone() method, which uses the clone() method contained in the Cloneable interface to create a duplicate of the cloned object. The PrototypeManagerAndDuplicator is used to provide access to Dragon object creation, since the Dragon constructor is private.

Simple Factory: There is a Pizza variable that is set to null because the specific type of pizza will be determined based on a parameter entered. Based on the type parameter entered in the createPizza(String type) method, the pizza variable will be set to a newly created, more specialized type of Pizza object (CheesePizza, PepperoniPizza, ClamPizza, VeggiePizza) and returned.