**Singleton:**

* Open Demos>Java>singleton in IntelliJ and navigate to src>java>main>Logger.java
* Walk students through what the file is and its methods
* In Main, create a new instance of Logger and call logWithdraw on it
* Run the program then check the log file, it should have logged the withdraw
* In Main, create a new logger and use it to log a transfer
* Run the program then check the log file, it should only have a transfer logged (everytime we create a logger we clear and re-write the file)
* In Logger, make the constructor private
* Add a private static Logger field called logger and initialize it to null
* Create a getInstance method that checks to see if an instance exist before making one and always returns the logger field
* Run the program, the log file should now have both the withdrawal and the transfer logged

**Factory Method:**

* Open Demos>Java>factory-method in IntelliJ and navigate to src>main>java
* Show the students the abstract Room class and its concrete implementations MagicRoom and OrdinaryRoom
  + These are the abstract product class and the concrete product classes that we saw on the previous slide
* Create a new abstract class called MazeGame, this is our abstract creator class from the last slide
  + Give it an ArrayList of Rooms field
  + Create a constructor that takes no arguments, creates 2 Rooms via a makeRoom() method
  + Call connect on the first room and pass in the second room
  + Add both rooms to the rooms ArrayList
  + Create an abstract method called makeRoom that returns a Room, this is our factory method
* Create a class called MagicMazeGame that extends MazeGame, this is one of our concrete creator classes from the last slide
  + Implement the makeRoom method to return an instance of MagicRoom
* Create a class called OrdinaryMazeGame that extends MazeGame, this is one of our concrete creator classes from the last slide
  + Implement the makeRoom method to return an instance of OrdinaryRoom

**Prototype:**

* ​Open Demos>Java>prototype in IntelliJ and navigate to src>main>java
* Create a new abstract class called Creature
  + It will implement the Cloneable interface (which allows us to override Object.clone() without receiving the CloneNotSupportedException)
  + Create a field for health and initialize it to 10
  + Override the clone method to return a creature, return super.clone() cast to a Creature, add the CloneNotSupportedException to the method signature
  + Create a public method called takeDamage that lower’s the creature’s health by 3 and returns void
* Create a new Zombie class that extends Creature and has no code of its own
* In Main, create an ArrayList of Creatures and instantiate a Zombie to add to the list
* In a try/catch block add 2 cloned Zombies to the list of creatures, and catch the CloneNotSupportedException
* Call takeDamage() on the zombie that was originally created
* Create a copy of the above try/catch block
* Loop through the list of creatures and print their health – the 1st zombie will have 7, the 2nd & 3rd will have 10, and the last 2 will also have 7.