**Inner Classes:**

* Open Demos>Java>inner-classes in IntelliJ and navigate to src>main>java
* Create a class named Outer
* Create a method called outerMethod() that returns void and takes no parameters
* In the body of outerMethod() create an inner class called Inner
* In Inner, create a method called innerMethod() that prints “This is called from the Inner class”
* Still in the body of outerMethod but after the definition of the Inner class, create an instance of the Inner class and call the innerMethod with it
* Create a static Main class within the Outer class
* Create a main method which instantiates Outer and calls the outerMethod
* Run the program
* Create an anonymous inner class in the main method that prints “This is called from the anonymous inner class” when outerMethod is called on it.
  + Outer anonOuter = new Outer() {  
     void outerMethod() {  
     System.out.println("This is called from the anonymous inner class");  
     }  
    };

**Enums:**

* Open Demos>Java>enums in IntelliJ and navigate to src>main>java
* Create a new enum type called Planet
* Add the planets to the list of constants
  + Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune
* Create a field for surfaceGravity
* Create a constructor that sets the surfaceGravity field– this will cause errors on all of the constants b/c they have been defined without surfaceGravity
* Add the surfaceGravity of each planet:
  + MERCURY(0.38), VENUS(0.91), EARTH(1.0), MARS(0.38), JUPITER(2.34), SATURN(0.93), URANUS(0.92), NEPTUNE(1.12);
* Create a getSurfaceWeight method that takes the double weightInPounds and returns weightInPounds \* this. surfaceGravity
* Open Main.java and, in the main method, create a Scanner
* Print text asking the user to input their weight in pounds, and create a variable to hold the users input as a double
* For each planet in Planet.values() format print "Your weight on %s is %.2f pounds%n", p, p.getSurfaceWeight(earthWeightInPounds)

**Arrays:**

* Open Demos>Java>arrays in IntelliJ and navigate to src>main>java>Main
* Create a Scanner for users to input numbers to get their sum
* Create a new int array with 5 elements
* Use a for loop to take input from the user and append it to the array until the limit of the array is reached
* Create a sum variable and use a for loop to add each number in the array to the sum
* Print the sum
* Run the program and enter 5 random numbers

**Time:**

* Open Demos>Java>date-time in IntelliJ and navigate to src>main>java>Main.java
* Open RoboFriend.java and walk students through what the methods in that class are doing
* Run the program and ask it about the date, time, and date-time
* Quit the program
* Create a new class called Formatter
  + Create a method called formatDate() that returns a DateTimeFormatter
  + formatDate should return DateTimeFormatter.ofPattern(“MM-dd-yyyy”)
* Add a formatter object to the RoboFriend class’ properties
* In RoboFriend, update the getDate method:
  + After LocalDate.now() chain on the format() method and pass in formatter.formatDate()
  + The getDate method should now return a String
* Run the program and ask to see the new date format
* In Formatter, create a formatTime method that uses the pattern “HH:mm”
* Also create a formatDateTime method that uses the pattern “HH:mm E, MMM dd yyyy«
* Run the program and request to see the new time and date time formats