This code demonstrates the **Prototype Design Pattern**, which is used to create new objects by copying or cloning existing objects. Instead of creating new instances from scratch, the prototype pattern allows creating new instances by cloning pre-existing prototypes, making object creation more efficient, especially for complex objects.

**Class Breakdown:**

1. **Shape Interface (Prototype Interface)**:
   * The Shape interface extends the Cloneable interface, signaling that classes implementing Shape can be cloned.
   * It declares two methods:
     + Shape clone(): Used to create a copy of the object.
     + void draw(): Abstract method for drawing the shape, implemented by concrete subclasses.
2. **Circle Class (Concrete Prototype)**:
   * Implements the Shape interface.
   * **clone() method**:
     + The clone() method creates and returns a copy of the Circle object using the super.clone() method from the Object class.
     + The try-catch block handles the CloneNotSupportedException if cloning isn't possible, although the exception is unlikely in this case because Cloneable is implemented.
   * **draw() method**: Prints "Drawing a Circle" to the console, simulating the drawing of a circle.
3. **Rectangle Class (Concrete Prototype)**:
   * Similar to the Circle class, this class implements the Shape interface.
   * **clone() method**: Creates a clone of the Rectangle object using the same logic as the Circle class.
   * **draw() method**: Prints "Drawing a Rectangle", representing the rectangle being drawn.
4. **PrototypePatternDemo (Client)**:
   * This is the **client class** that demonstrates the usage of the Prototype pattern.
   * **Object Cloning**:
     + A Circle object (circlePrototype) is created, and a new instance (clonedCircle) is obtained by calling clone() on the prototype. The draw() method is then called on the cloned object, resulting in the output "Drawing a Circle."