public interface ConnectionPool {  
 /\*\*  
 \* Returns a connection from this pool if it is available  
 \* or throws SQLException when no connection available  
 \* **@return** connection from this pool  
 \* **@throws** SQLException thrown when connection is not available  
 \*/  
 Connection getConnection() throws SQLException, InterruptedException;  
  
 /\*\*  
 \* Returns a connection from this pool if it is available  
 \* otherwise waits for no more than timeout milliseconds to get a connection  
 \* **@param** timeout timeout in milliseconds  
 \* **@return** connection from this pool is it becomes available within timeout milliseconds  
 \* **@throws** SQLException thrown when connection is not available  
 \*/  
 Connection getConnection(long timeout) throws SQLException, InterruptedException;  
  
 /\*\*  
 \* Returns connection to the pool  
 \* **@param** connection connection to be returned to the pool  
 \*/  
 void releaseConnection(Connection connection);  
}

import java.sql.Connection;  
import java.sql.Driver;  
import java.sql.SQLException;  
import java.util.LinkedList;  
import java.util.Queue;  
  
  
public class ConnectionPoolImpl implements ConnectionPool {  
 private Driver driver;  
 private String jdbcUrl;  
 private String username;  
 private String password;  
 private int maximumPoolSize;  
 private int size;  
 private Queue<Connection> connections;  
  
 public ConnectionPoolImpl(String driverClassName, String jdbcUrl,  
 String username, String password, int maximumPoolSize)  
 throws ClassNotFoundException, IllegalAccessException, InstantiationException {  
 Class c = Class.forName(driverClassName);  
 this.driver = (Driver) c.newInstance();  
  
 this.jdbcUrl = jdbcUrl;  
 this.username = username;  
 this.password = password;  
 this.maximumPoolSize = maximumPoolSize;  
 this.size = 0;  
 this.connections = new LinkedList<>();  
 }  
  
 @Override  
 public Connection getConnection() throws SQLException, InterruptedException {  
 return getConnection(0);  
 }  
  
 private Connection createNewConnection() throws SQLException {  
 try {  
 java.util.Properties info = new java.util.Properties();  
 info.put("user", username);  
 info.put("password", password);  
  
 return driver.connect(jdbcUrl, info);  
 } catch (Throwable t) {  
 synchronized (this) {  
 size--;  
 this.notifyAll();  
 }  
 t.printStackTrace();  
 throw new SQLException("Connection not available", t);  
 }  
 }  
  
 @Override  
 public Connection getConnection(long timeout) throws SQLException, InterruptedException {  
 long timestamp = System.currentTimeMillis() + timeout;  
  
 boolean createNewConnection = false;  
  
 synchronized (this) {  
 while (connections.isEmpty()) {  
 if (size < maximumPoolSize) {  
 size++;  
 createNewConnection = true;  
 break;  
 } else {  
 this.wait(Math.max(timestamp - System.currentTimeMillis(), 1));  
   
 if (timestamp <= System.currentTimeMillis()) {  
 throw new SQLException("Connection not available");  
 }   
 }  
 }  
  
 if (!createNewConnection) {  
 return connections.poll();  
 }  
 }  
  
 return createNewConnection();  
 }  
  
 @Override  
 public void releaseConnection(Connection connection) {  
 synchronized (this) {  
 connections.offer(connection);  
 this.notifyAll();  
 }  
 }  
}