Design Patterns Project
UE18CS341

Team 26
Nikita Joanne - PES1201800808
Sneha Hegde - PES1201801157

Project Title
Connection Pool Implementation (Generalised)

Design Patterns Used
- Object Pool (primary)
- Factory (secondary)

Programming Language
Java

Abstract
In a typical web application server, there will be many threads servicing user requests. The data needed to service these requests may be in a database. The thread will have to:

1. Open Connection
2. Run Query
3. Process query response
4. Send response to user

Opening a connection to a database is an expensive operation - often taking multiple seconds. Moreover, each database has a limit on the maximum number of connections it can have open at a time.

The Connection Pool (an implementation of the Object Pool Pattern) is used to overcome these limitations. The Pool object will maintain a list of "live" connections. The user thread simply gets a connection from the pool (saving time on step 1 above). After running the query, it will release the connection back to the pool.
Some additional complexities in Connection Pool:

- Automatically "Expand" the Pool size as we run out of connections - as the number of available connections comes down, we want to automatically increase the pool size to avoid running out of Connections

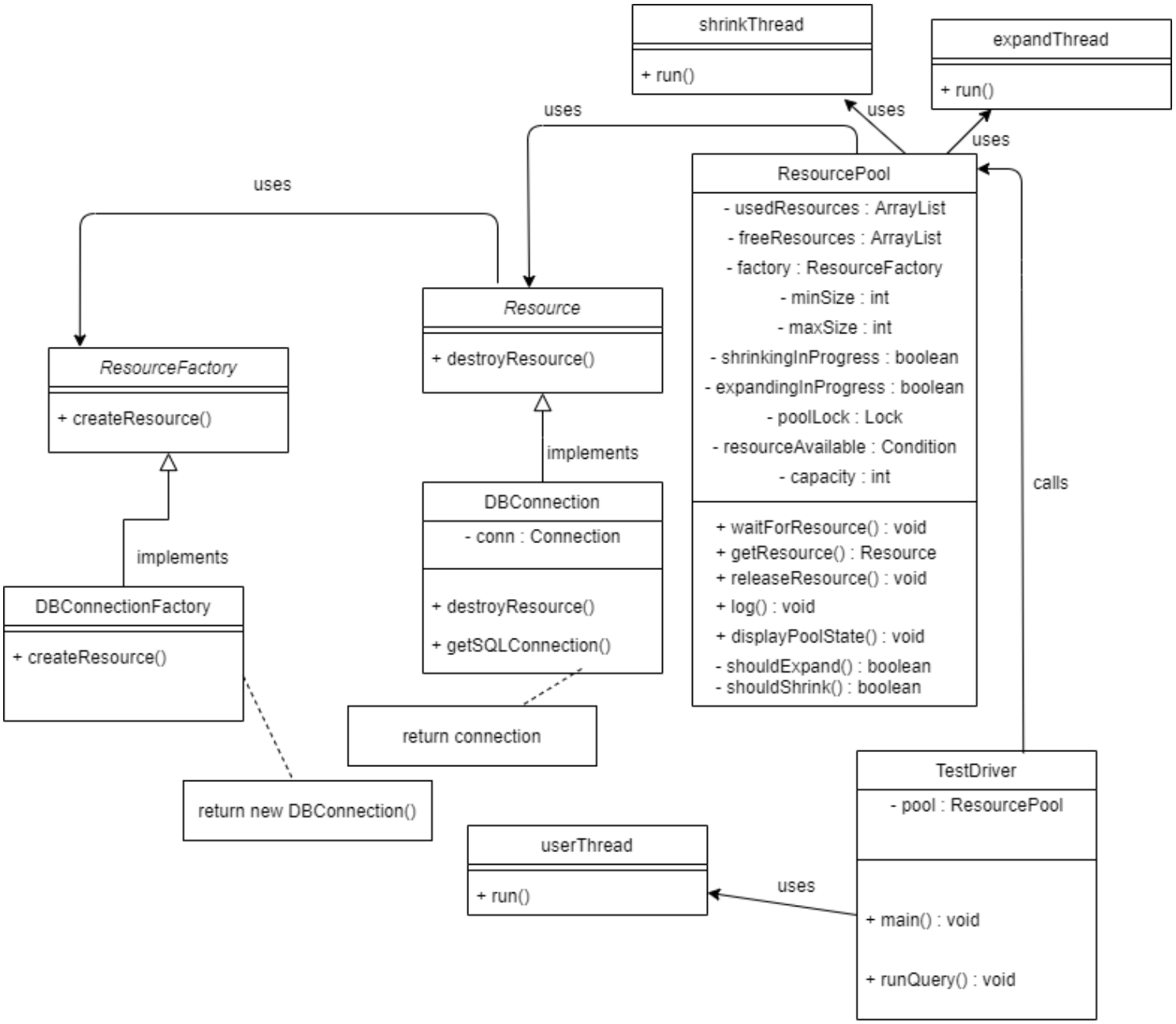
- Automatically "Shrink" the pool size - to reduce the load on the database, the Connection Pool can reduce the number of connections in the pool

Scope for Learning

Listing out learning points that have been covered in earlier courses, but never applied in a practical sense. This project affords us a chance to learn through implementation.

- Threads (thread usage and safety)
- DB connection and query handling
- Experience coding in Java

UML Diagram



Implementation Details

ResourcePool.java

- This is the program which creates and manages the resource pool.
- It maintains two arraylists, one with all the free resources, and one with all the used resources.
- It facilitates the acquiring and releasing of resources.
- Based on the percentage of free connections, it can expand or shrink the pool. It uses threads to do so, so that expansion and shrinkage can happen independently, without having to stall the user.
- Utilizes threads to expand and shrink the size of the pool, to prevent stall of execution
- A locking mechanism is employed to prevent race conditions

Resource.java

- Interface for managing resources
- In our project, we have used this for destroying a resource upon completion

ResourceFactory.java

- Interface to generate resources using factory pattern
- Declares a method createResource, which is overridden in the interface implementation

DBConnection.java

- Implements the Resource interface
- Forms and returns a connection to the MySQL database
- Severs the connection upon completion of the query

DBConnectionFactory.java

- Implements the ResourceFactory interface
- Returns a connection to a MySQL database

TestDriver.java

- Uses threading to simulate multiple users accessing resources simultaneously
- Each user thread runs five queries before terminating the connection to the database
- After the connection is terminated, the thread sleeps for a random period, to simulate the system waiting for a new user

Execution Prerequisites

- MySQL installed
- JDBC installed

Screenshots (javadoc and output)

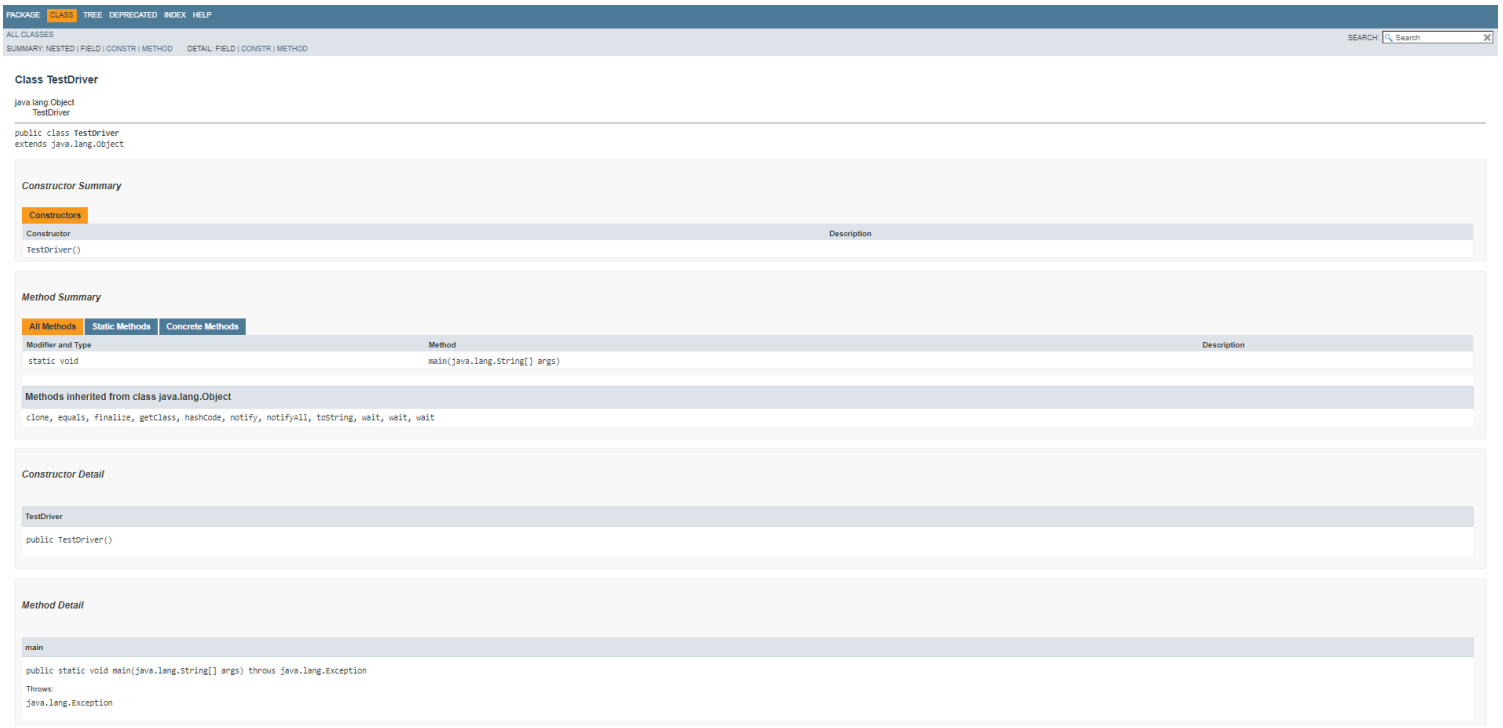
I. Output

```
------------------------------
Begin: Thread-46
Expanded by: 21
End: Thread-46
------------------------------


------------------------------
Begin: Thread-46
***** Resource Pool State *****
Used Size: 19
Free Size: 22
******************************
End: Thread-46
------------------------------


------------------------------
Begin: Thread-45
Shrunk by: 7
End: Thread-45
------------------------------


------------------------------
Begin: Thread-45
***** Resource Pool State *****
Used Size: 21
Free Size: 14
******************************
End: Thread-45
------------------------------


------------------------------
Begin: Thread-53
Shrunk by: 5
End: Thread-53
------------------------------
```

```
------------------------------
Begin: Thread-53
***** Resource Pool State *****
Used Size: 21
Free Size: 9
******************************
End: Thread-53
------------------------------


------------------------------
Begin: Thread-54
Shrunk by: 3
End: Thread-54
------------------------------


------------------------------
Begin: Thread-54
***** Resource Pool State *****
Used Size: 21
Free Size: 6
******************************
End: Thread-54
------------------------------


------------------------------
Begin: Thread-56
Shrunk by: 2
End: Thread-56
------------------------------


------------------------------
Begin: Thread-56
***** Resource Pool State *****
Used Size: 21
Free Size: 4
******************************
End: Thread-56
------------------------------
```

II. Javadoc:

1. TestDriver.java



PACKAGE CLASS TREE DEPRECATED INDEX HELP

ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

SEARCH: Search

Class TestDriver

java.lang.Object
  TestDriver

```
public class TestDriver
extends java.lang.object
```

Constructor Summary

Constructors

Constructor	Description
TestDriver()	

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method	Description
static void	main(java.lang.String[] args)	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TestDriver

```
public TestDriver()
```

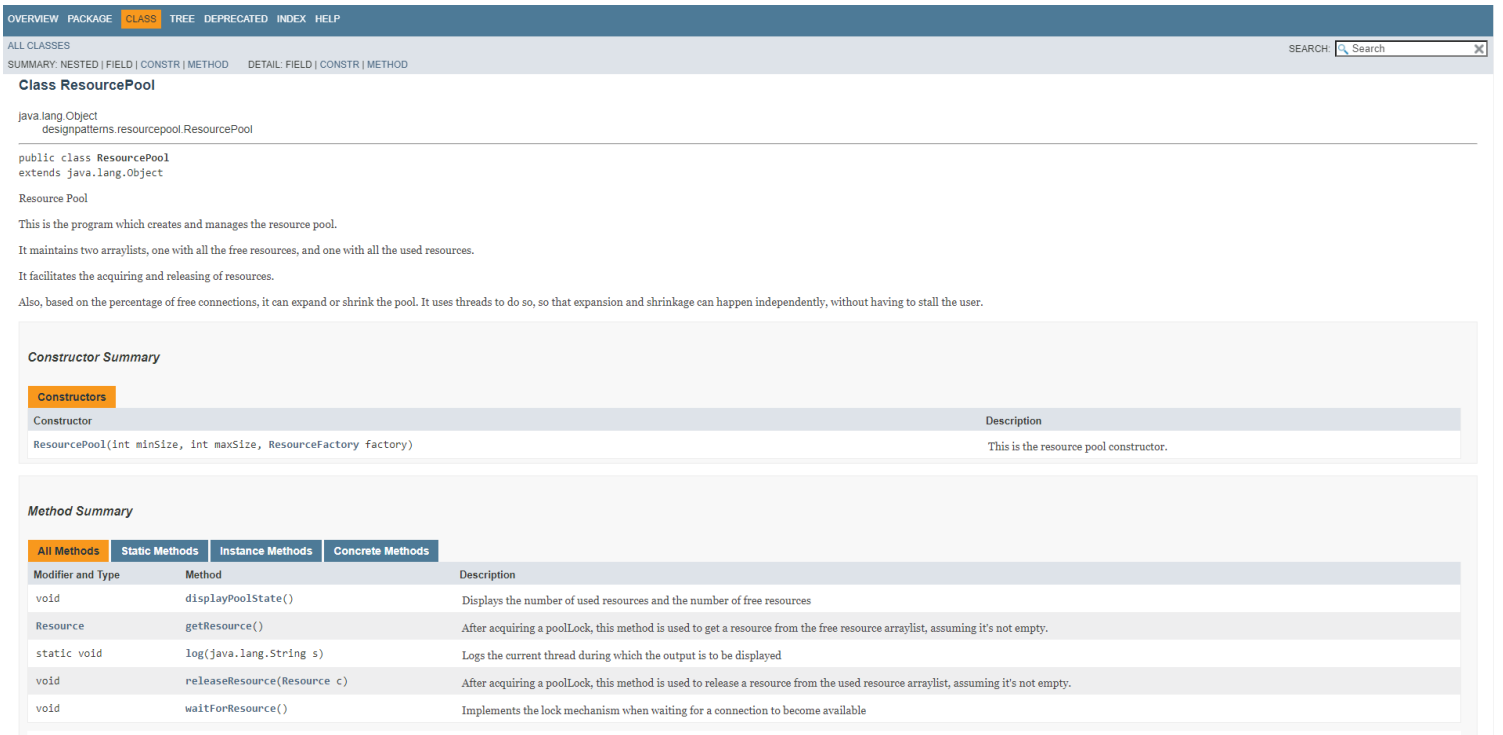
Method Detail

main

```
public static void main(java.lang.String[] args) throws java.lang.Exception
```

Throws:
java.lang.Exception

2. ResourcePool.java



OVERVIEW PACKAGE CLASS TREE DEPRECATED INDEX HELP

ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

SEARCH: Search

Class ResourcePool

java.lang.Object
  designpatterns.resourcepool.ResourcePool

```
public class ResourcePool
extends java.lang.Object
```

Resource Pool

This is the program which creates and manages the resource pool.

It maintains two arraylists, one with all the free resources, and one with all the used resources.

It facilitates the acquiring and releasing of resources.

Also, based on the percentage of free connections, it can expand or shrink the pool. It uses threads to do so, so that expansion and shrinkage can happen independently, without having to stall the user.

Constructor Summary

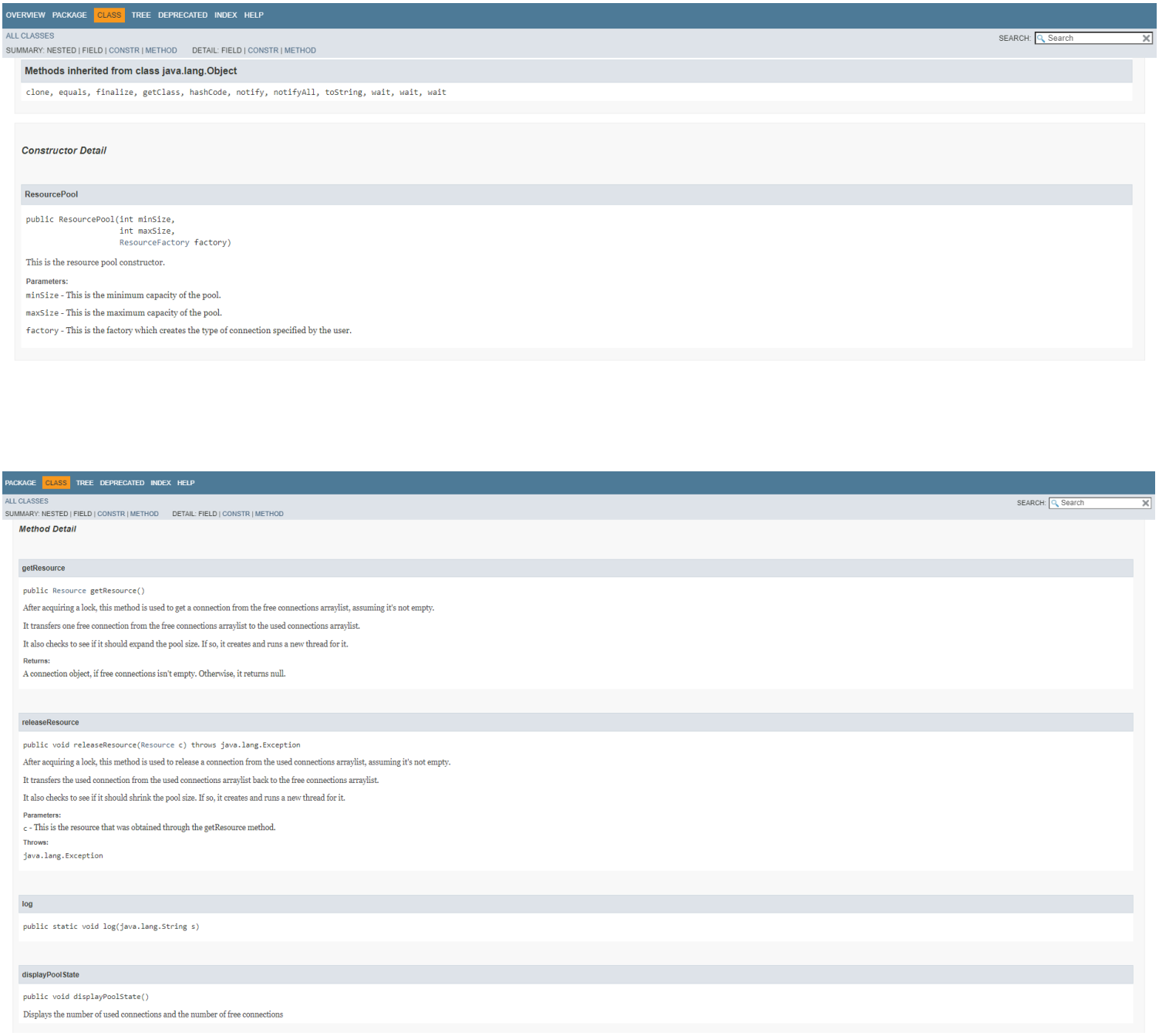
Constructors

Constructor	Description
ResourcePool(int minSize, int maxSize, ResourceFactory factory)	This is the resource pool constructor.

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
void	displayPoolState()	Displays the number of used resources and the number of free resources
Resource	getResource()	After acquiring a poolLock, this method is used to get a resource from the free resource arraylist, assuming it's not empty.
static void	log(java.lang.String s)	Logs the current thread during which the output is to be displayed
void	releaseResource(Resource c)	After acquiring a poolLock, this method is used to release a resource from the used resource arraylist, assuming it's not empty.
void	waitForResource()	Implements the lock mechanism when waiting for a connection to become available



OVERVIEW PACKAGE CLASS TREE DEPRECATED INDEX HELP

ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

SEARCH: Search

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ResourcePool

```
public ResourcePool(int minSize,
                    int maxSize,
                    ResourceFactory factory)
```

This is the resource pool constructor.

Parameters:

`minSize` - This is the minimum capacity of the pool.

`maxSize` - This is the maximum capacity of the pool.

`factory` - This is the factory which creates the type of connection specified by the user.

PACKAGE CLASS TREE DEPRECATED INDEX HELP

ALL CLASSES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

SEARCH: Search

Method Detail

getResource

```
public Resource getResource()
```

After acquiring a lock, this method is used to get a connection from the free connections arraylist, assuming it's not empty.

It transfers one free connection from the free connections arraylist to the used connections arraylist.

It also checks to see if it should expand the pool size. If so, it creates and runs a new thread for it.

Returns:

A connection object, if free connections isn't empty. Otherwise, it returns null.

releaseResource

```
public void releaseResource(Resource c) throws java.lang.Exception
```

After acquiring a lock, this method is used to release a connection from the used connections arraylist, assuming it's not empty.

It transfers the used connection from the used connections arraylist back to the free connections arraylist.

It also checks to see if it should shrink the pool size. If so, it creates and runs a new thread for it.

Parameters:

`c` - This is the resource that was obtained through the getResource method.

Throws:

`java.lang.Exception`

log

```
public static void log(java.lang.String s)
```

displayPoolState

```
public void displayPoolState()
```

Displays the number of used connections and the number of free connections

Future Scope/Improvements

- Incorporate singleton pattern (there only needs to be one resource pool object)
- Due to multiple threads running, the order of output is in slight disarray. To improve presentation, the output could be redirected and ordered for improved readability.

References

- Object Pool Pattern - https://www.oodesign.com/object-pool-pattern.html
- Locks - Condition (Java Platform SE 7) (oracle.com)
- MySQL and Java JDBC Tutorial - https://www.vogella.com/tutorials/MySQLJava/article.html
- Javadoc - https://www.tutorialspoint.com/java/java_documentation.htm

Conclusion

The initial objective of this project was to implement a connection pool. The implementation was generalized to allow for various other resources to be used with the resource pool The factory design pattern was employed in the implementation. The project solidified concepts such as:

- Object Oriented Programming
- Database Connections through JDBC
- Threads and thread safety
- Usage of the javadoc tool to generate documentation for the code