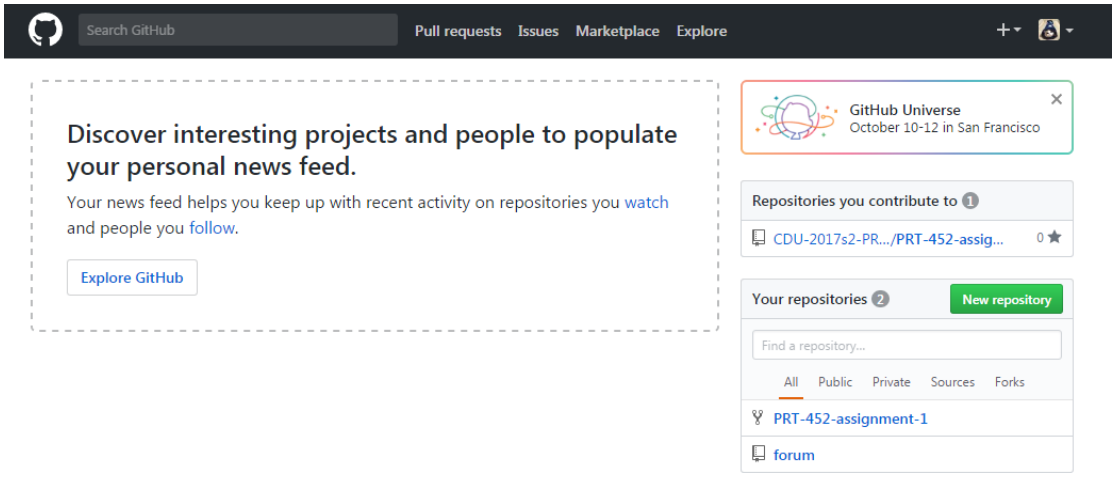
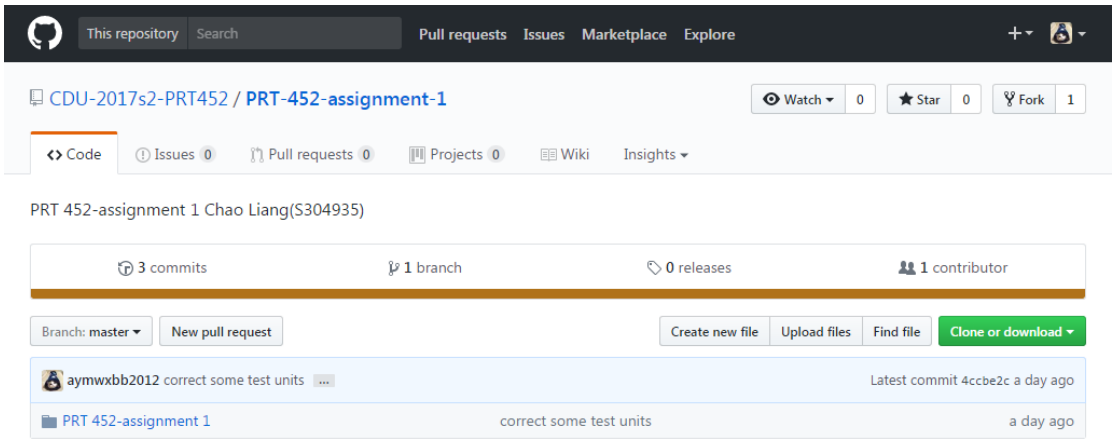
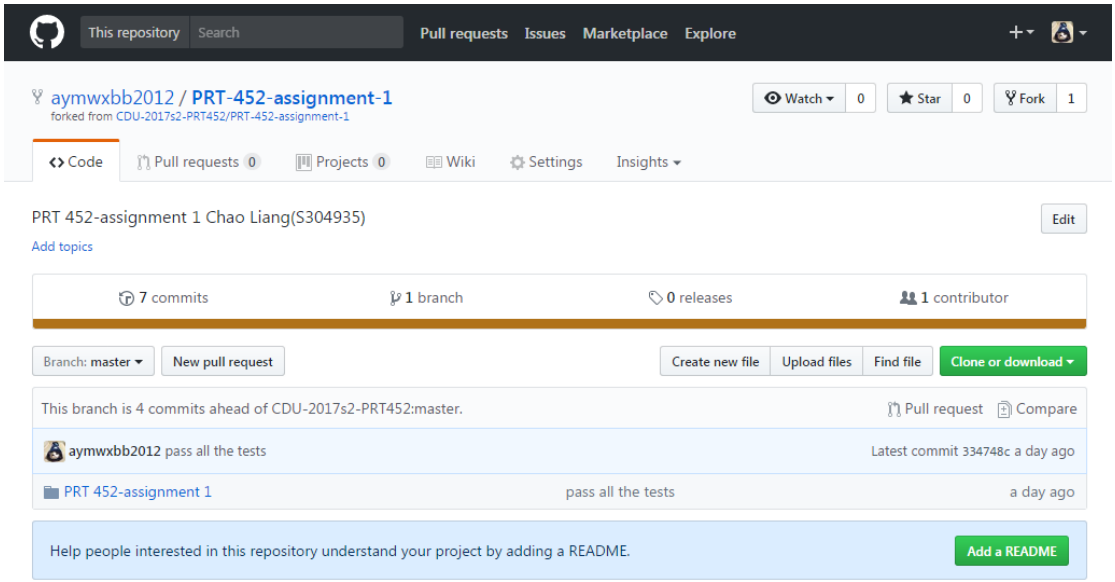
Question1

Answer:





At first, I submitted my assignment (including programming code) into this organization repository.

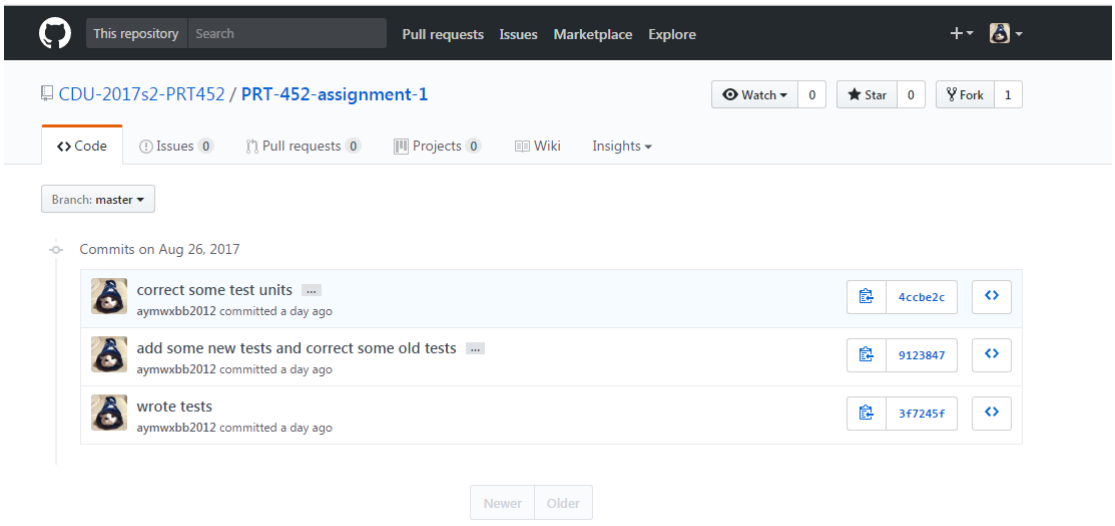


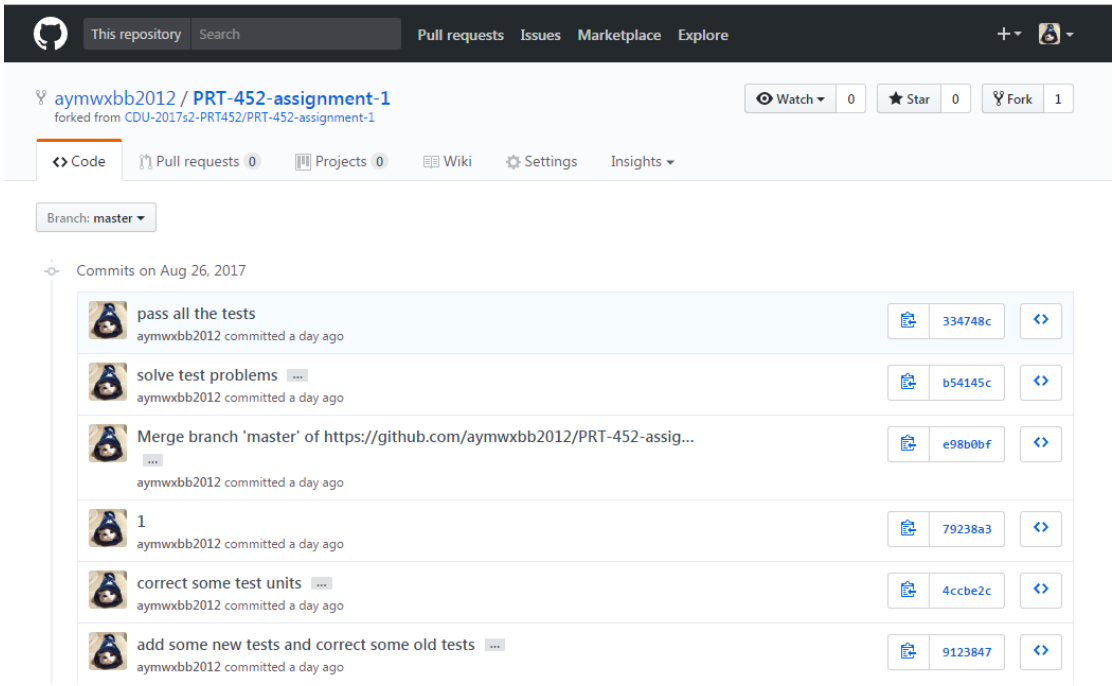
Then, I fork them into my own repository and finished them.

https://github.com/aymwxbb2012/PRT-452-assignment-1

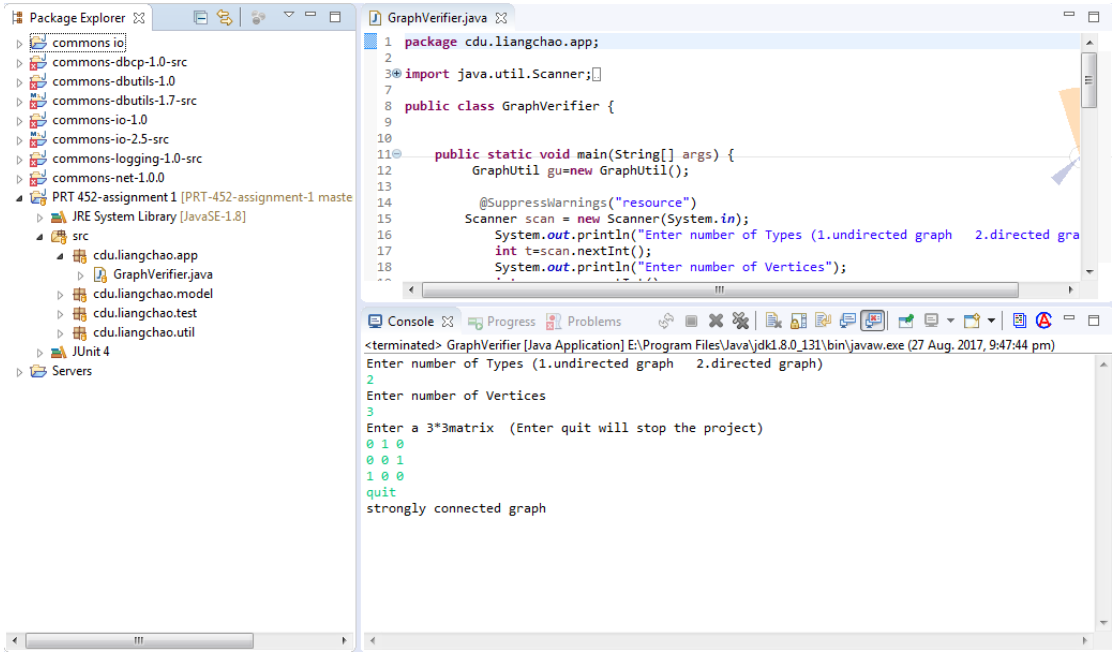
Question2

Answer:





https://github.com/aymwxbb2012/PRT-452-assignment-1



Question3

Answer:

Example 1: Switch statements

commons-io-2.5-src/src/main/java/org/apache/commons/io/FileSystemUtils.java

```
long freeSpaceOS(final String path, final int os, final boolean kb, final
long timeout) throws IOException {
        if (path == null) {
            throw new IllegalArgumentException("Path must not be null");
        }
        switch (os) {
            case WINDOWS:
                return kb ? freeSpaceWindows(path, timeout) /
FileUtils.ONE_KB : freeSpaceWindows(path, timeout);
            case UNIX:
                return freeSpaceUnix(path, kb, false, timeout);
            case POSIX_UNIX:
                return freeSpaceUnix(path, kb, true, timeout);
            case OTHER:
                throw new IllegalStateException("Unsupported operating
system");
            default:
                throw new IllegalStateException(
                  "Exception caught when determining operating system");
        }
    }
```

```
50
51      /** Operating system state flag for error. */
52      private static final int INIT_PROBLEM = -1;
53      /** Operating system state flag for neither Unix nor Windows. */
54      private static final int OTHER = 0;
55      /** Operating system state flag for Windows. */
56      private static final int WINDOWS = 1;
57      /** Operating system state flag for Unix. */
58      private static final int UNIX = 2;
59      /** Operating system state flag for Posix flavour Unix. */
60      private static final int POSIX_UNIX = 3;
61
62      /** The operating system flag. */
63      private static final int OS;
64
```

From my point of view, this is switch statements. “WINDOWS”, “UNIX”, “POSIX_ UNIX” and “OTHER” are a list of type code. When a new condition is added, people have to find all the switch code and modify it. People can think of polymorphism and use Replace Type Code with Subclasses or Replace Type Code with State/Strategy to modify it. Because there are not too many conditions, polymorphism will be superfluous. People can separate this method into multiple smaller methods by using Replace Parameter with Explicit Methods and change the switch accordingly.

Example 2: Data clumps

/commons-io-1.0/src/java/org/apache/commons/io/EndianUtils.java

```
126     public static short readSwappedShort( byte[] data, int offset )
127     {
128         return (short)( ( ( data[ offset + 0 ] & 0xff ) << 0 ) +
129             ( ( data[ offset + 1 ] & 0xff ) << 8 ) );
130     }
131
132     /**
133      * Reads an unsigned short (16-bit) value from a byte array at a given
134      * offset. The value is converted to the opposed endian system while
135      * reading.
136      * @param data source byte array
137      * @param offset starting offset in the byte array
138      * @return the value read
139      */
140     public static int readSwappedUnsignedShort( byte[] data, int offset )
141     {
142         return (int)( ( ( data[ offset + 0 ] & 0xff ) << 0 ) +
143             ( ( data[ offset + 1 ] & 0xff ) << 8 ) );
144     }
145
146     /**
147      * Writes a "int" value to a byte array at a given offset. The value is
148      * converted to the opposed endian system while writing.
149      * @param data target byte array
150      * @param offset starting offset in the byte array
151      * @param value value to write
152      */
```

```
168     public static int readSwappedInteger( byte[] data, int offset )
169     {
170         return (int)( ( ( data[ offset + 0 ] & 0xff ) << 0 ) +
171             ( ( data[ offset + 1 ] & 0xff ) << 8 ) +
172             ( ( data[ offset + 2 ] & 0xff ) << 16 ) +
173             ( ( data[ offset + 3 ] & 0xff ) << 24 ) );
174     }
175
176     /**
177      * Reads an unsigned integer (32-bit) value from a byte array at a given
178      * offset. The value is converted to the opposed endian system while
179      * reading.
180      * @param data source byte array
181      * @param offset starting offset in the byte array
182      * @return the value read
183      */
184     public static long readSwappedUnsignedInteger( byte[] data, int offset )
185     {
186         return (long)( ( ( data[ offset + 0 ] & 0xff ) << 0 ) +
187             ( ( data[ offset + 1 ] & 0xff ) << 8 ) +
188             ( ( data[ offset + 2 ] & 0xff ) << 16 ) +
189             ( ( data[ offset + 3 ] & 0xff ) << 24 ) );
190     }
191
```

```
252     public static float readSwappedFloat( byte[] data, int offset )
253     {
254         return Float.intBitsToFloat( readSwappedInteger( data, offset ) );
255     }
256
257     /**
258      * Writes a "double" value to a byte array at a given offset. The value is
259      * converted to the opposed endian system while writing.
260      * @param data target byte array
261      * @param offset starting offset in the byte array
262      * @param value value to write
263      */
264     public static void writeSwappedDouble( byte[] data, int offset, double value )
265     {
266         writeSwappedLong( data, offset, Double.doubleToLongBits( value ) );
267     }
268
269     /**
270      * Reads a "double" value from a byte array at a given offset. The value is
271      * converted to the opposed endian system while reading.
272      * @param data source byte array
273      * @param offset starting offset in the byte array
274      * @return the value read
275      */
276     public static double readSwappedDouble( byte[] data, int offset )
277     {
278         return Double.longBitsToDouble( readSwappedLong( data, offset ) );
279     }
280
```

I think that this is data clumps. "**byte**[] data" and "**int** offset" often appear together in the code. People can use Introduce Parameter Object to set them off and become a new class so as to improve understanding and organization of code and reduce the size of code. But if people move only these parameters to a new class, this will become a Data Class.

Example 3: Message chains

/commons-dbcp-1.0-src/src/java/org/apache/commons/dbcp/PoolingDriver.java

```
synchronized public ObjectPool getPool(String name) {
        ObjectPool pool = (ObjectPool)(_pools.get(name));
        if(null == pool) {
            InputStream in =
this.getClass().getResourceAsStream(String.valueOf(name) + ".jocl");
            if(null != in) {
                JOCLContentHandler jocl = null;
                try {
                    jocl = JOCLContentHandler.parse(in);
                } catch(Exception e) {
                    throw new DbcpException(e);
                }
```

```
                if(jocl.getType(0).equals(String.class)) {
                    pool = getPool((String)(jocl.getValue(0)));
                    if(null != pool) {
                        registerPool(name,pool);
                    }
                } else {
                    pool =
((PoolableConnectionFactory)(jocl.getValue(0))).getPool();
                    if(null != pool) {
                        registerPool(name,pool);
                    }
                }
            }
        }
        return pool;
    }
```

This is a message chain. JOCLContentHandler requests PoolableConnectionFactory, and PoolableConnectionFactory requests Pool. This means that JOCLContentHandler has to rely on the navigation between PoolableConnectionFactory and Pool. If any modification happens in these relationships, people have to change JOCLContentHandler. People can reduce the message chain by using Hide Delegate. By deleting message chain, the client code does not need to know too many details of the relationship of objects and the code size can be reduced. However, if people use too many Hide Delegate, understanding the functionality and operation of the code will become difficult, and may cause the Middle Men problem.

Example 4: Speculative generality

Commons NetNET-242:

Method createServerSocket of FTPSSocketFactory will never be called and thus UseClientMode is incorrect in a secured ftp transfer using active mode.

https://issues.apache.org/jira/browse/NET-242?jql=project%20%3D%20NET%20AND%20status%20in%20(Resolved%2C%20Closed)%20AND%20text%20~%20%22delete%22%20ORDER%20BY%20key%20DESC

Index: FTPSSocketFactory.java
===

```
--- FTPSSocketFactory.java     (revision 712101)
+++ FTPSSocketFactory.java     (working copy)
@@ -20,13 +20,11 @@

 import java.io.IOException;
 import java.net.InetAddress;
-import java.net.ServerSocket;
 import java.net.Socket;
 import java.net.UnknownHostException;

 import javax.net.SocketFactory;
 import javax.net.ssl.SSLContext;
-import javax.net.ssl.SSLServerSocket;


 /**
@@ -63,20 +61,5 @@
         return this.context.getSocketFactory().createSocket(address,
port, localAddress, localPort);
     }

-    public ServerSocket createServerSocket(int port) throws
IOException {
-        return
this.init(this.context.getServerSocketFactory().createServerSocket(po
rt));
-    }
-
-    public ServerSocket createServerSocket(int port, int backlog)
throws IOException {
-        return
this.init(this.context.getServerSocketFactory().createServerSocket(po
rt, backlog));
-    }
-
-    public ServerSocket createServerSocket(int port, int backlog,
InetAddress ifAddress) throws IOException {
-        return
this.init(this.context.getServerSocketFactory().createServerSocket(po
rt, backlog, ifAddress));
-    }
-
-    public ServerSocket init(ServerSocket socket) throws IOException
{
```

```
-         ((SSLServerSocket) socket).setUseClientMode(true);
-         return socket;
-     }
+
  }
```

This bug is speculative generality. According to the description in the web site, the method createServerSocket is never used, because people can create ServerSockets by using a ServerSocketFactory. People can use Inline Method to get rid of unused class, methods and parameters, or just delete them.

Example 5: Speculative generality

Commons NetNET-621

SubnetUtils#SubnetInfo - remove unnecessary accessors

https://issues.apache.org/jira/browse/NET-621?jql=project%20in%20(DBCP%2C%20IO%2C%20DBUTILS%2C%20LOGGING%2C%20NET)%20AND%20text%20~%20%22remove%20unnecessary%22

```
95
96          private SubnetInfo() {}
97
98  -       private int netmask()       { return netmask; }
99  -       private int network()       { return network; }
100 -       private int address()       { return address; }
101 -       private int broadcast()     { return broadcast; }
102 -
103         // long versions of the values (as unsigned int) which are more suitable for
    range checking
104         private long networkLong()  { return network &  UNSIGNED_INT_MASK; }
105         private long broadcastLong(){ return broadcast &  UNSIGNED_INT_MASK; }
```

```
95
96          private SubnetInfo() {}
97
98          // long versions of the values (as unsigned int) which are more suitable for
    range checking
99          private long networkLong()  { return network &  UNSIGNED_INT_MASK; }
100         private long broadcastLong(){ return broadcast &  UNSIGNED_INT_MASK; }
101
102         private int low() {
103 +           return (isInclusiveHostCount() ? network :
104 +               broadcastLong() - networkLong() > 1 ? network + 1 : 0);
105         }
106
```

This is Speculative generality. According to the description in the web site, The methods, including “private int netmask() { return netmask; }”,”private int network() { return network; }”,”private int address() { return address; }”,”private int broadcast() { return broadcast; }”, do not actually work. These methods is created for the future, but not necessary. These methods will make the code hard to understand and support.

References:

FOWLER, M., BECK, K., BRANT, J., OPDYKE, W. & ROBERTS, D. (1999) Refactoring: Improving the Design of Existing Code, Addison Wesley.

GAMMA, E., HELM, R., JOHNSON, R. & VLISSIDES, J. (1995) Design patterns : elements of reusable object-oriented software, Reading, Mass., Addison-Wesley.