Pacemaker Solution (Assignment 1)

Produced by:

Dr. Siobhán Drohan (sdrohan@wit.ie)

Eamonn de Leastar (edeleastar@wit.ie)

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

https://www.eclipse.org/papyrus/

In eclipse, help, install new software:
http://download.eclipse.org/modeling/mdt/papyrus/updates/releases/oxygen

Initialising a new maven project in Eclipse

```
C:\Users\Siobhan\Desktop\agile workspace\pacemaker-console-solution>dir

19/10/2017  14:45    <DIR>          .
19/10/2017  14:45    <DIR>          ..
19/10/2017  14:33             1,122 pom.xml
19/10/2017  14:45    <DIR>          src
               1 File(s)          1,122 bytes
               3 Dir(s)  60,339,650,560 bytes free
```

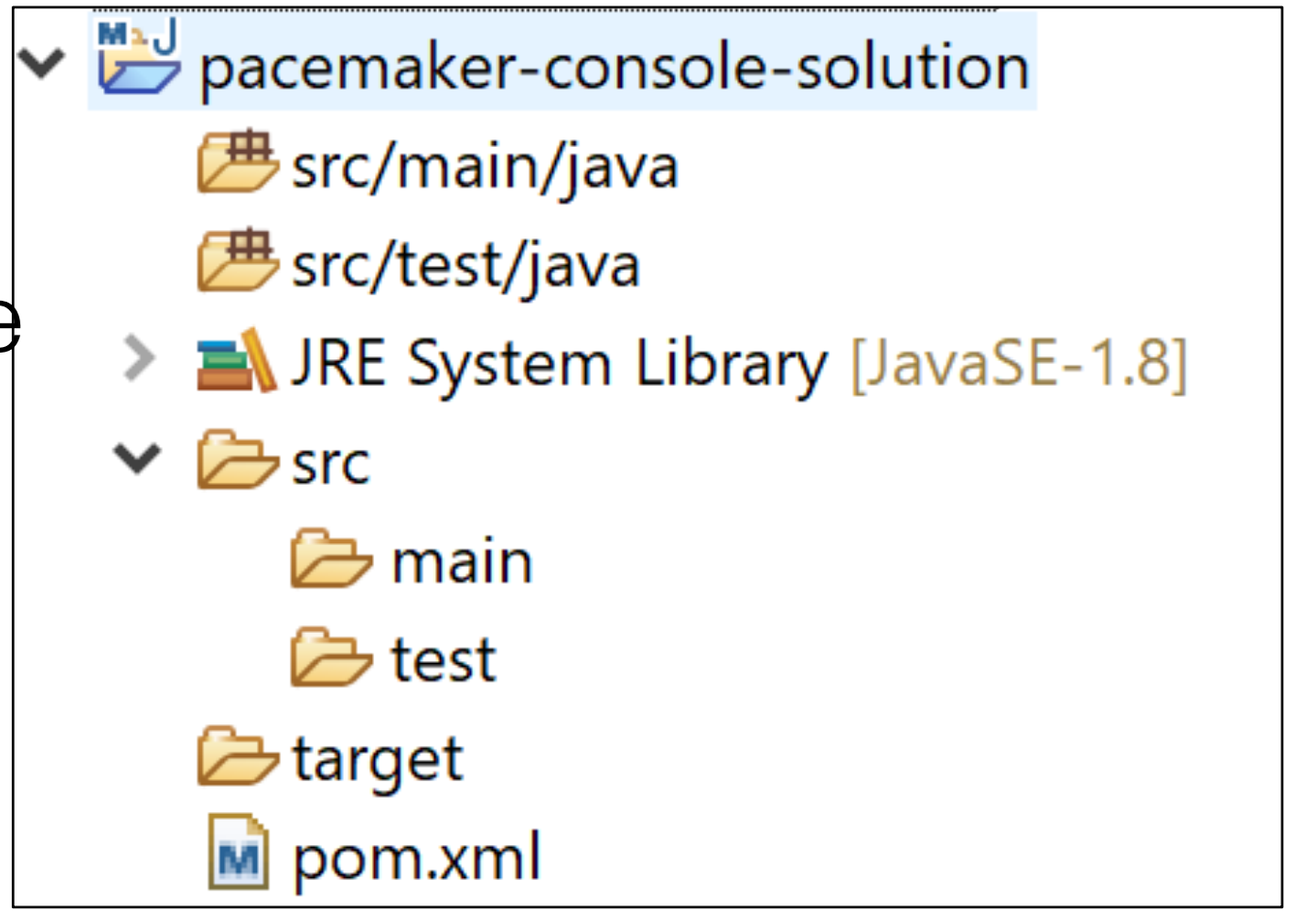
Create an empty folder in your workspace and add this pom.xml to it.

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>pacemaker</groupId>
  <artifactId>pacemaker-console-solution</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>pacemaker-console-solution</name>
  <url>https://wit-computing-msc-2017.github.io/agile</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <version>3.7.0</version>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>

  <dependencies>
  </dependencies>
</project>
```

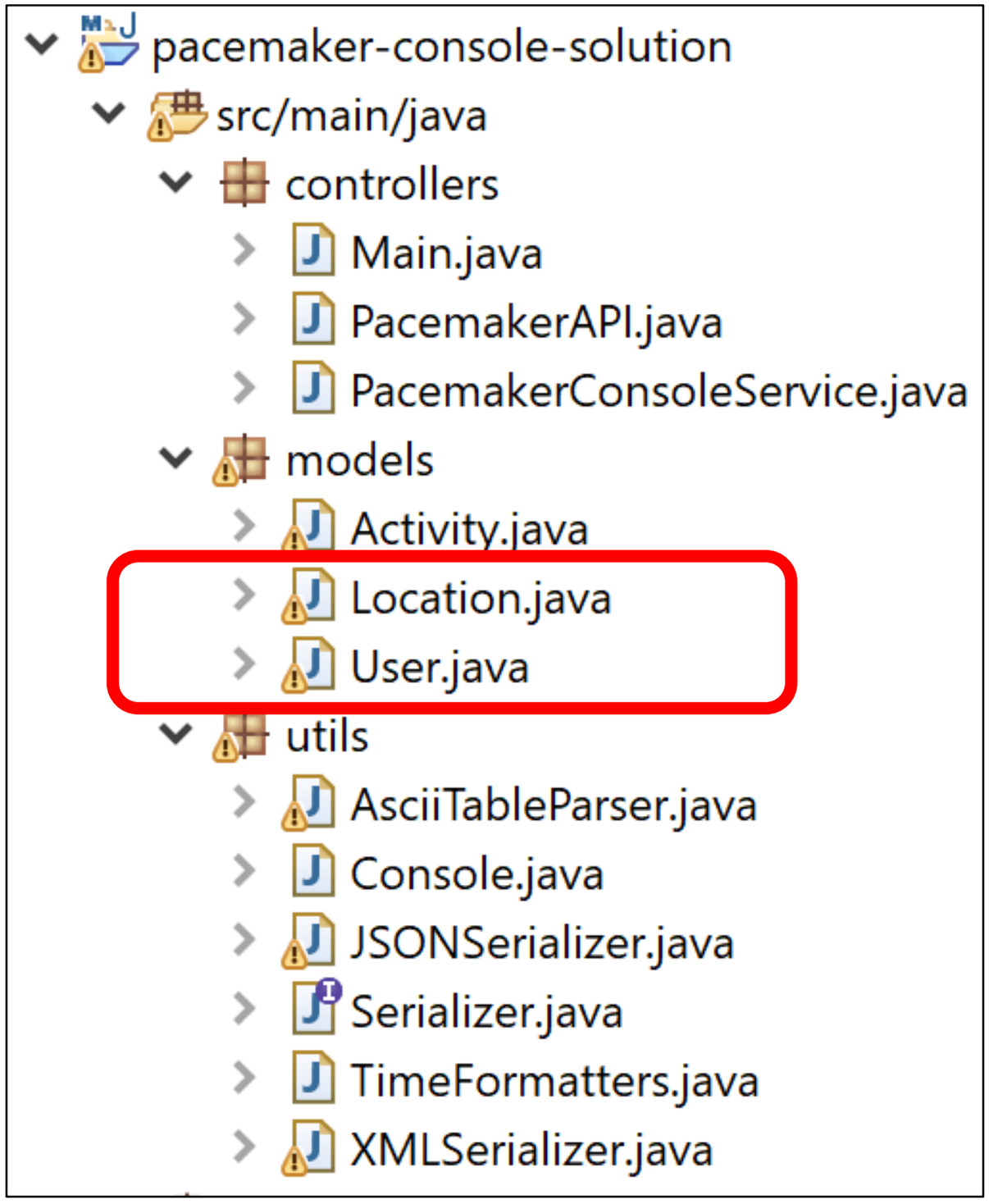
```
C:\Users\Siobhan\Desktop\agile workspace\pacemaker-console-solution>mvn validate
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building pacemaker-console-solution 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.158 s
[INFO] Finished at: 2017-10-19T14:46:58+01:00
[INFO] Final Memory: 5M/123M
[INFO] ------------------------------------------------------------------------

C:\Users\Siobhan\Desktop\agile workspace\pacemaker-console-solution>
```

Once your pom.xml validates, open Eclipse and perform:

Import… → Existing Maven Projects



User and Location Models



```
package models;

import static com.google.common.base.MoreObjects.toStringHelper;
import java.io.Serializable;
import java.util.HashMap;
import java.util.Map;
import java.util.UUID;
import com.google.common.base.Objects;

public class User implements Serializable {

  public String id;
  public String firstName;
  public String lastName;
  public String email;
  public String password;

  public Map<String, Activity> activities = new HashMap<>();

  public User() {
  }

  public String getId() {
    return id;
  }

  public String getFirstname() {
    return firstName;
  }

  public String getLastname() {
    return lastName;
  }

  public String getEmail() {
    return email;
  }
```

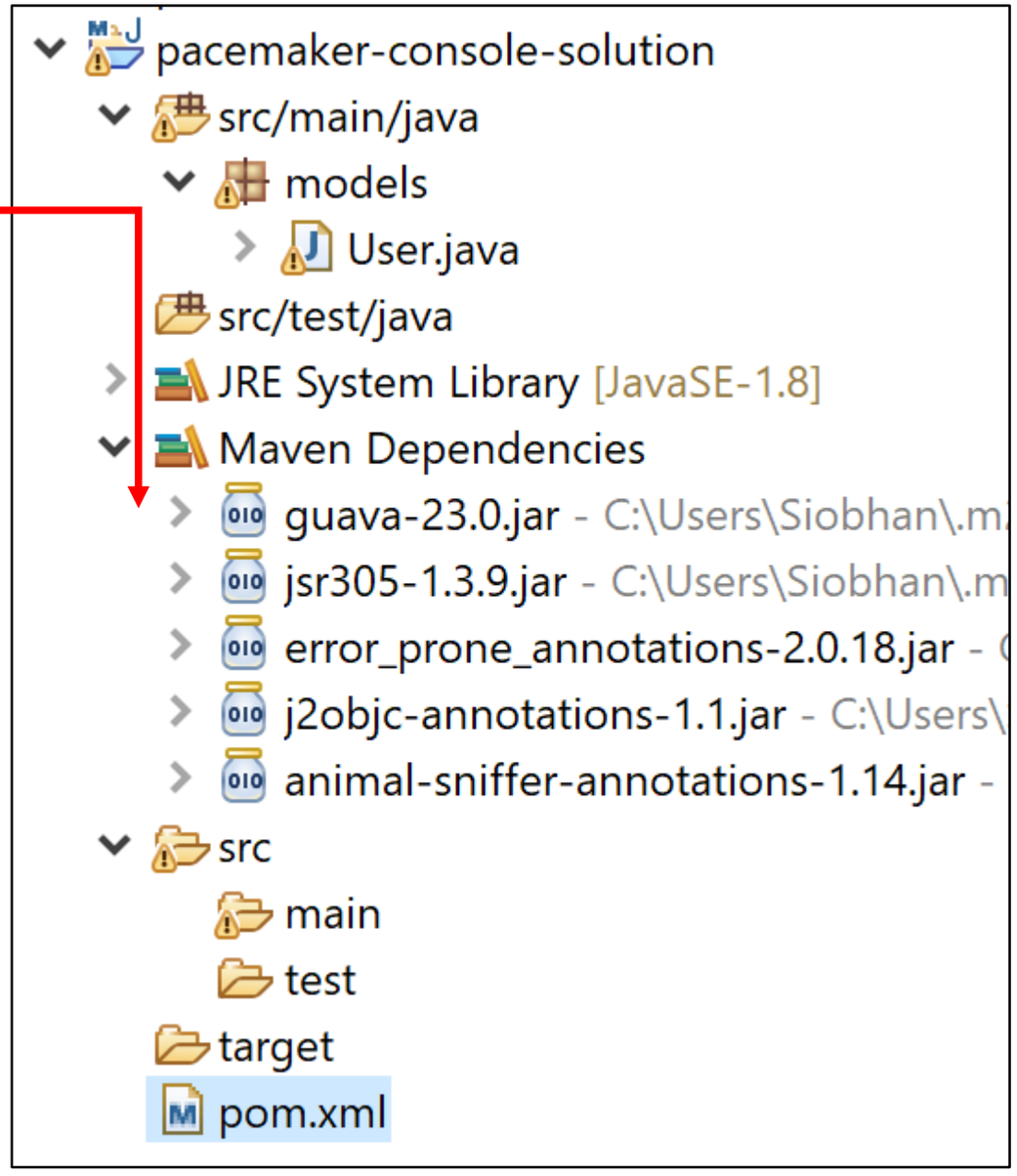
Guava is not on the build path, so com.google.common.* isn't recognised.

```
  public User(String firstName, String lastName, String email, String pass
    this.id = UUID.randomUUID().toString();
    this.firstName = firstName;
    this.lastName = lastName;
    this.email = email;
    this.password = password;
  }

  @Override
  public boolean equals(final Object obj) {
    if (obj instanceof User) {
      final User other = (User) obj;
      return Objects.equal(firstName, other.firstName)
          && Objects.equal(lastName, other.lastName)
          && Objects.equal(email, other.email)
          && Objects.equal(password, other.password)
          && Objects.equal(activities, other.activities);
    } else {
      return false;
    }
  }

  @Override
  public String toString() {
    return toStringHelper(this).addValue(id)
        .addValue(firstName)
        .addValue(lastName)
        .addValue(password)
        .addValue(email)
        .addValue(activities)
        .toString();
  }

  @Override
  public int hashCode() {
    return Objects.hashCode(this.id, this.lastName, this.firstName,
                                        this.email, this.password,  this.activities);
  }
}
```

```xml
<dependency>
  <groupId>com.google.guava</groupId>
  <artifactId>guava</artifactId>
  <version>23.0</version>
</dependency>
```

Add the above dependency to the pom.xml.

Eclipse Oxygen triggers a maven build and adds Guava as a dependency
→ errors resolved.



```java
public class User implements Serializable {

  //fields omitted – no change to them
  public Map<String, Activity> activities = new HashMap<>();

  //accessors & constructor omitted – no change to them

  @Override
  public boolean equals(final Object obj) {
    if (obj instanceof User) {
      final User other = (User) obj;
      return Objects.equal(firstName, other.firstName)
          && Objects.equal(lastName, other.lastName)
          && Objects.equal(email, other.email)
          && Objects.equal(password, other.password)
          && Objects.equal(activities, other.activities);
    } else {
      return false;
    }
  }

  @Override
  public String toString() {
    return toStringHelper(this).addValue(id)
        .addValue(firstName)
        .addValue(lastName)
        .addValue(password)
        .addValue(email)
        .addValue(activities)
        .toString();
  }

  @Override
  public int hashCode() {
    return Objects.hashCode(this.id, this.lastName, this.firstName,
                            this.email, this.password,  this.activities);
  }
}
```

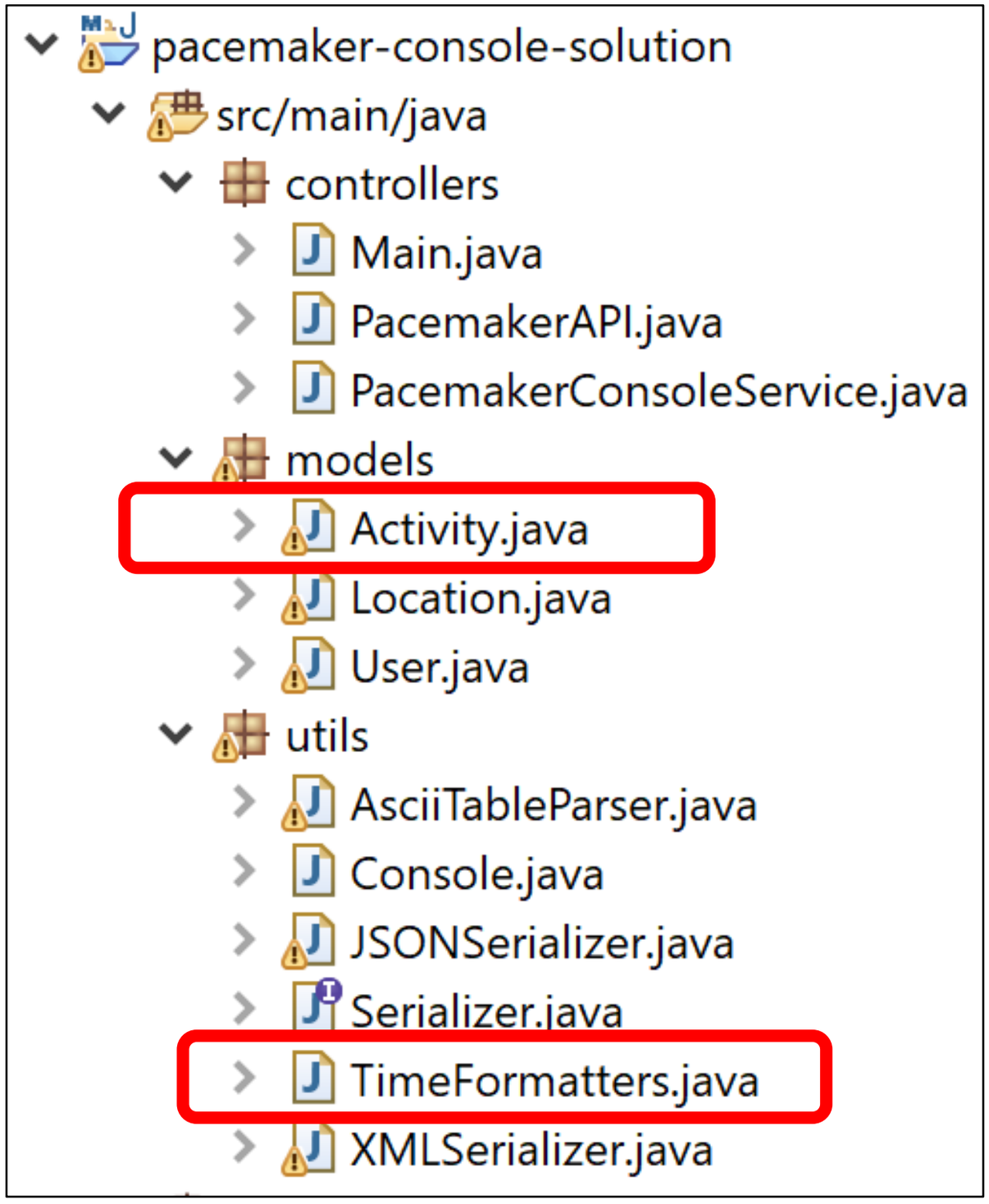
Notice that we refactored User to contain a HashMap of activities.

Relationship:
A User has zero to many Activity objects!

```
package models;

import static com.google.common.base.MoreObjects.toStringHelper;
import java.io.Serializable;
import java.util.UUID;
import com.google.common.base.Objects;

public class Location implements Serializable {

  public String id;
  public double longitude;
  public double latitude;

  public Location() {
  }

  public Location(double latitude, double longitude) {
    this.id = UUID.randomUUID().toString();
    this.latitude = latitude;
    this.longitude = longitude;
  }
```

```
  @Override
  public boolean equals(final Object obj) {
    if (obj instanceof Location) {
      final Location other = (Location) obj;
      return Objects.equal(latitude, other.latitude)
          && Objects.equal(longitude, other.longitude);
    } else {
      return false;
    }
  }

  @Override
  public String toString() {
    return toStringHelper(this).addValue(id)
        .addValue(latitude)
        .addValue(longitude)
        .toString();
  }

  @Override
  public int hashCode() {
    return Objects.hashCode(this.id, this.latitude,
                            this.longitude);
  }
}
```

Activity Model & TimeFormatters



TimeFormatter

```
package utils;

import org.joda.time.format.PeriodFormatterBuilder;
import org.joda.time.format.DateTimeFormat;
import org.joda.time.format.DateTimeFormatter;
import org.joda.time.format.PeriodFormatter;

import org.joda.time.DateTime;
import org.joda.time.Duration;

public class TimeFormatters {

  static PeriodFormatter periodFormatter = new PeriodFormatterBuilder().printZeroAlways()
      .appendHours()
      .appendSeparator(":")
      .appendMinutes()
      .appendSeparator(":")
      .appendSeconds()
      .toFormatter();

  static DateTimeFormatter dateFormatter = DateTimeFormat.forPattern("dd:MM:yyyy HH:mm:ss");

  public static DateTime parseDateTime(String dateTime) {
    return new DateTime(dateFormatter.parseDateTime(dateTime));
  }

  public static String parseDateTime(DateTime dateTime) {
    return dateFormatter.print(dateTime);
  }

  public static Duration parseDuration(String duration) {
    return periodFormatter.parsePeriod(duration).toStandardDuration();
  }

  public static String parseDuration(Duration duration) {
    return periodFormatter.print(duration.toPeriod());
  }
}
```

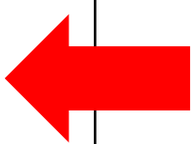
Joda-Time is not on the build path, so org.joda.* isn't recognised.

```
package models;

//imports omitted

import org.joda.time.DateTime;
import org.joda.time.Duration;
import static utils.TimeFormatters.parseDateTime;
import static utils.TimeFormatters.parseDuration;

public class Activity implements Serializable {

  //code omitted

  public DateTime starttime;
  public Duration duration;

  //code omitted

}
```

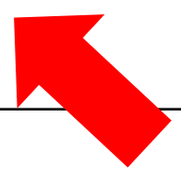
Include the Joda-Time dependency in the pom.xml

Refactor existing Activity to include date/time handling using our new utility.

```java
@Override
public String toString() {
  return toStringHelper(this).addValue(id)
      .addValue(type)
      .addValue(location)
      .addValue(distance)
      .addValue(parseDateTime(starttime))
      .addValue(parseDuration(duration))
      .addValue(route)
      .toString();
}
```

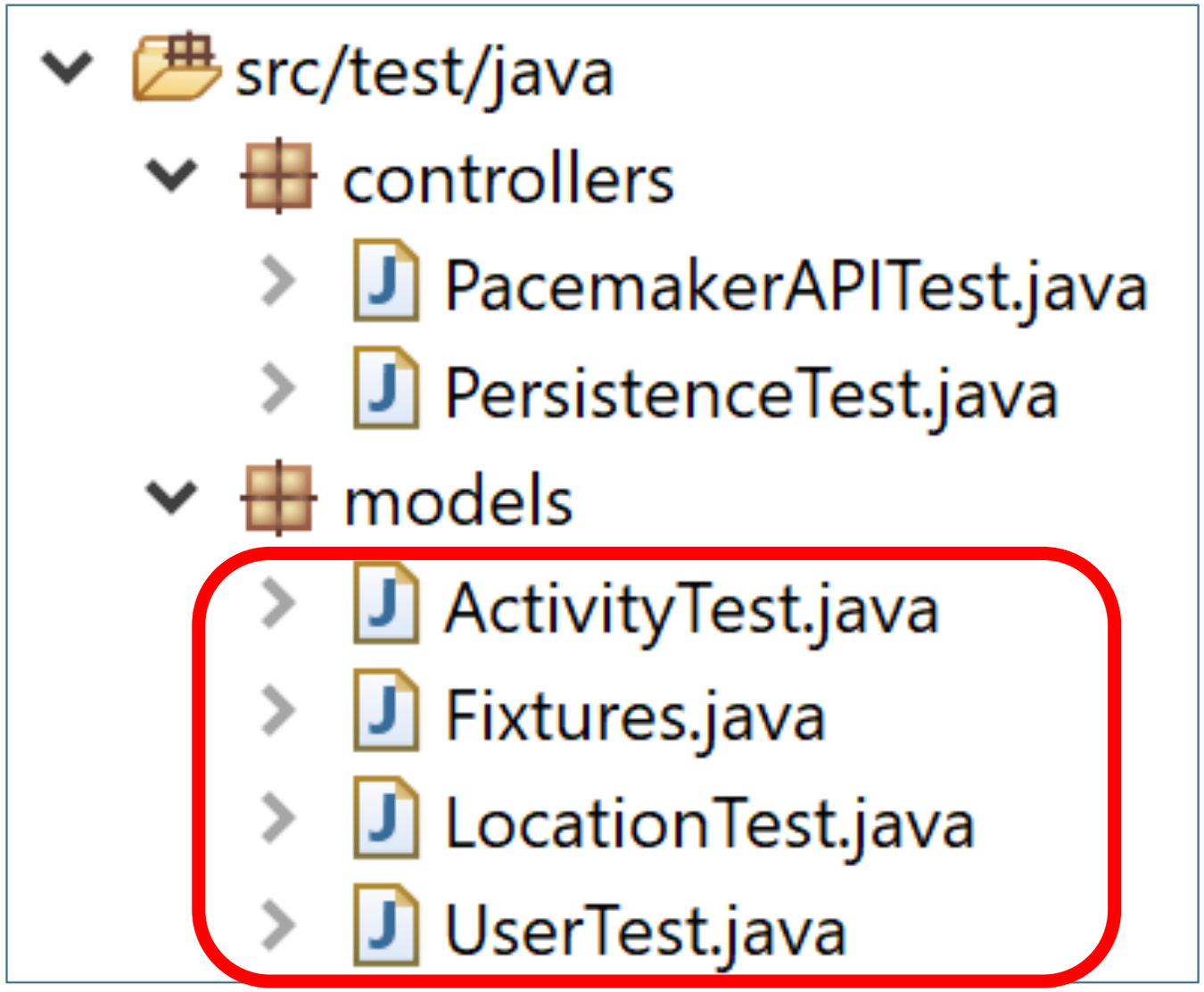


```java
public Activity(String type, String location, double distance,
                String start, String duration) {
    this.id = UUID.randomUUID().toString();
    this.type = type;
    this.location = location;
    this.distance = distance;
    this.starttime = parseDateTime(start);
    this.duration = parseDuration(duration);
  }
```



Refactor Activity to include date/time handling using our new utility.

Models Testing

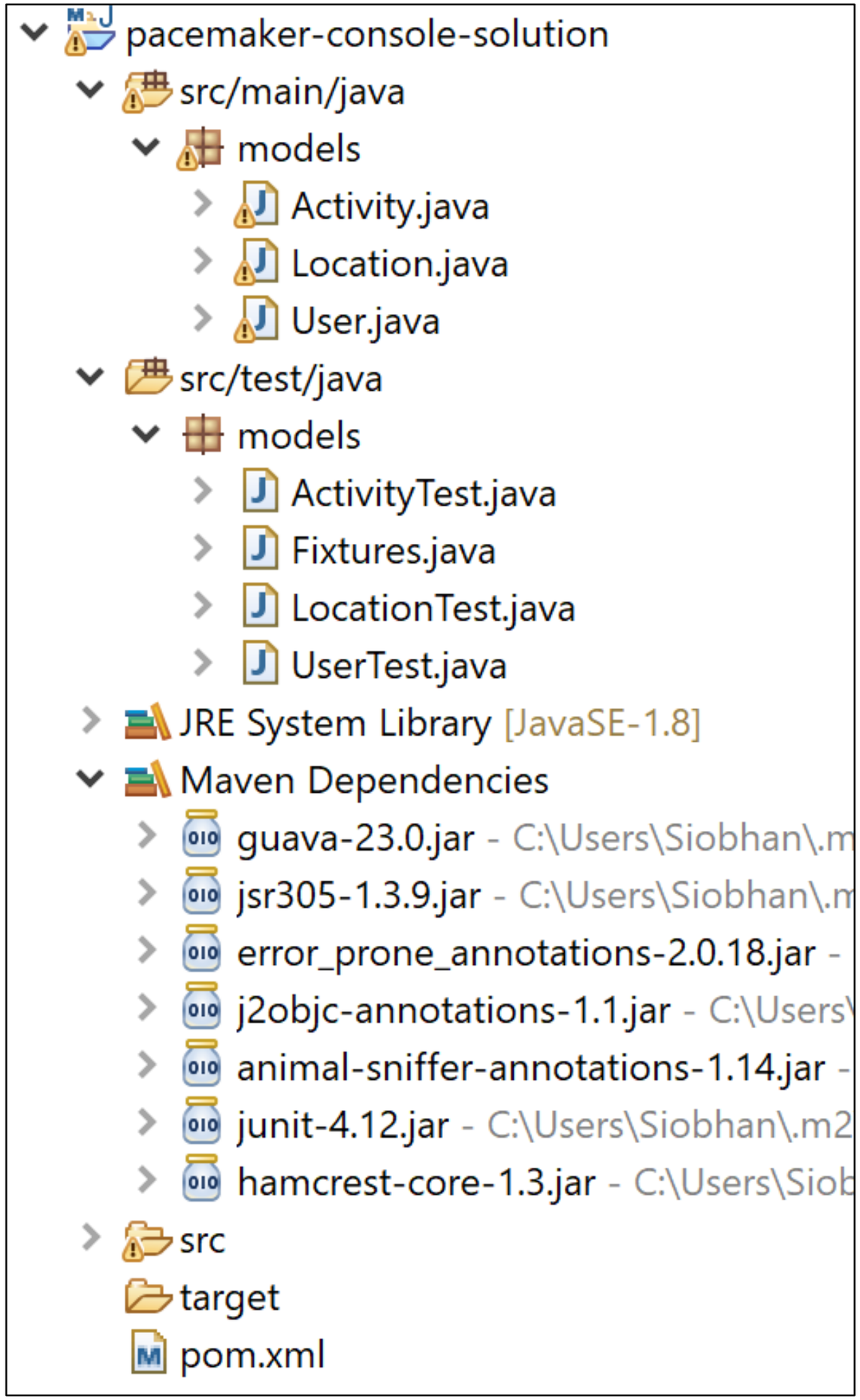


```java
package models;

import static org.junit.Assert.*;
import java.util.HashSet;
import java.util.Set;
import org.junit.Test;

import static models.Fixtures.users;

public class UserTest {

  User homer = new User("homer", "simpson", "homer@simpson.com", "secret");

  @Test
  public void testCreate() {
    assertEquals("homer", homer.firstName);
    assertEquals("simpson", homer.lastName);
    assertEquals("homer@simpson.com", homer.email);
    assertEquals("secret", homer.password);
  }

  @Test
  public void testIds() {
    Set<String> ids = new HashSet<>();
    for (User user : users) {
      ids.add(user.id);
    }
    assertEquals(users.size(), ids.size());
  }

  @Test
  public void testToString() {
    assertEquals("User{" + homer.id + ", homer, simpson, secret, homer@simpson.com}",
        homer.toString());
  }
}
```

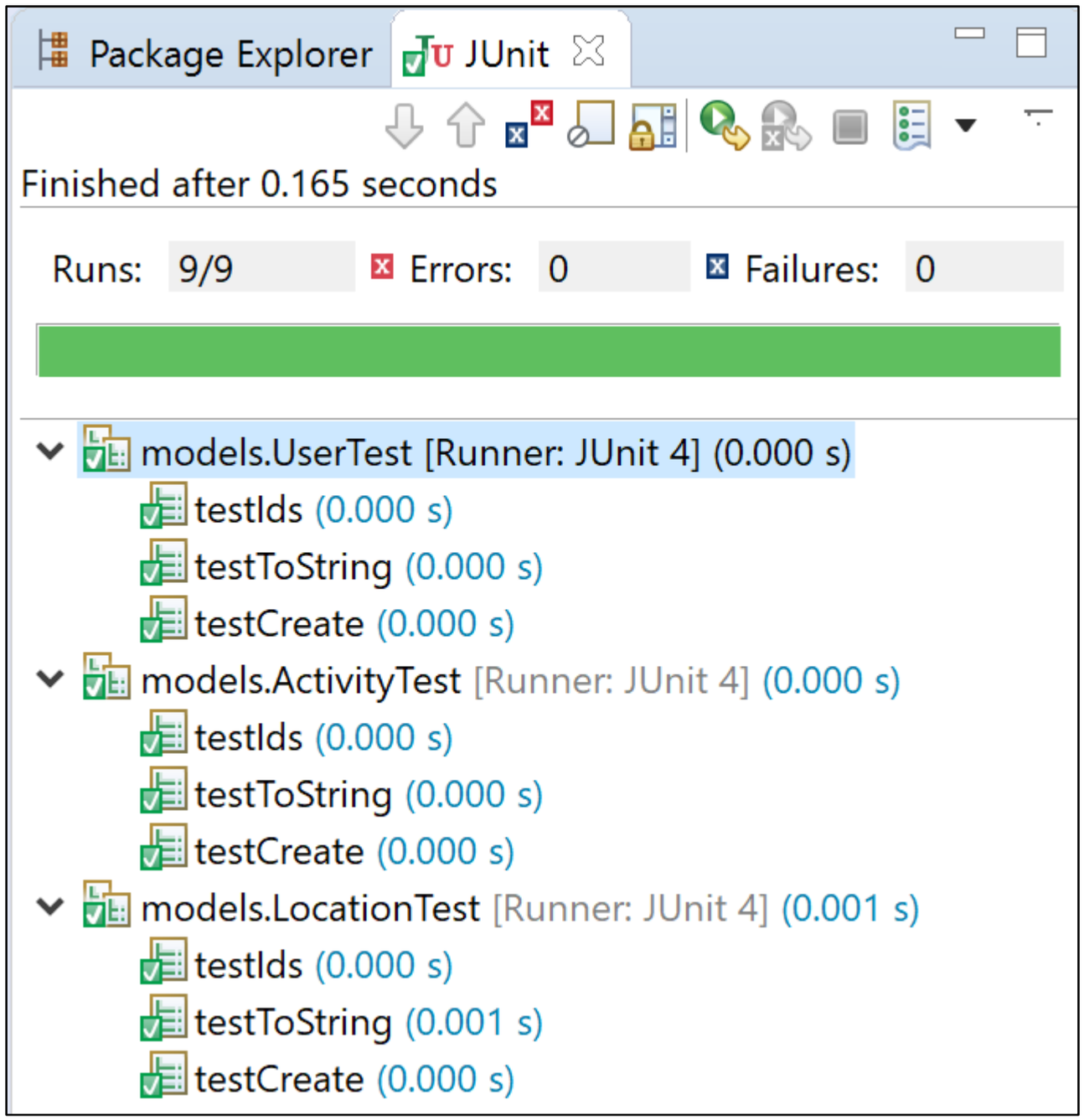
```java
package models;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Fixtures {

  public static List<User> users = new ArrayList<>(Arrays.asList(
      new User("marge", "simpson", "marge@simpson.com", "secret"),
      new User("lisa", "simpson", "lisa@simpson.com", "secret"),
      new User("bart", "simpson", "bart@simpson.com", "secret"),
      new User("maggie", "simpson", "maggie@simpson.com", "secret")));
}
```

JUnit is not on the build path, so org.junit.* isn't recognised.

Include JUnit in the pom.



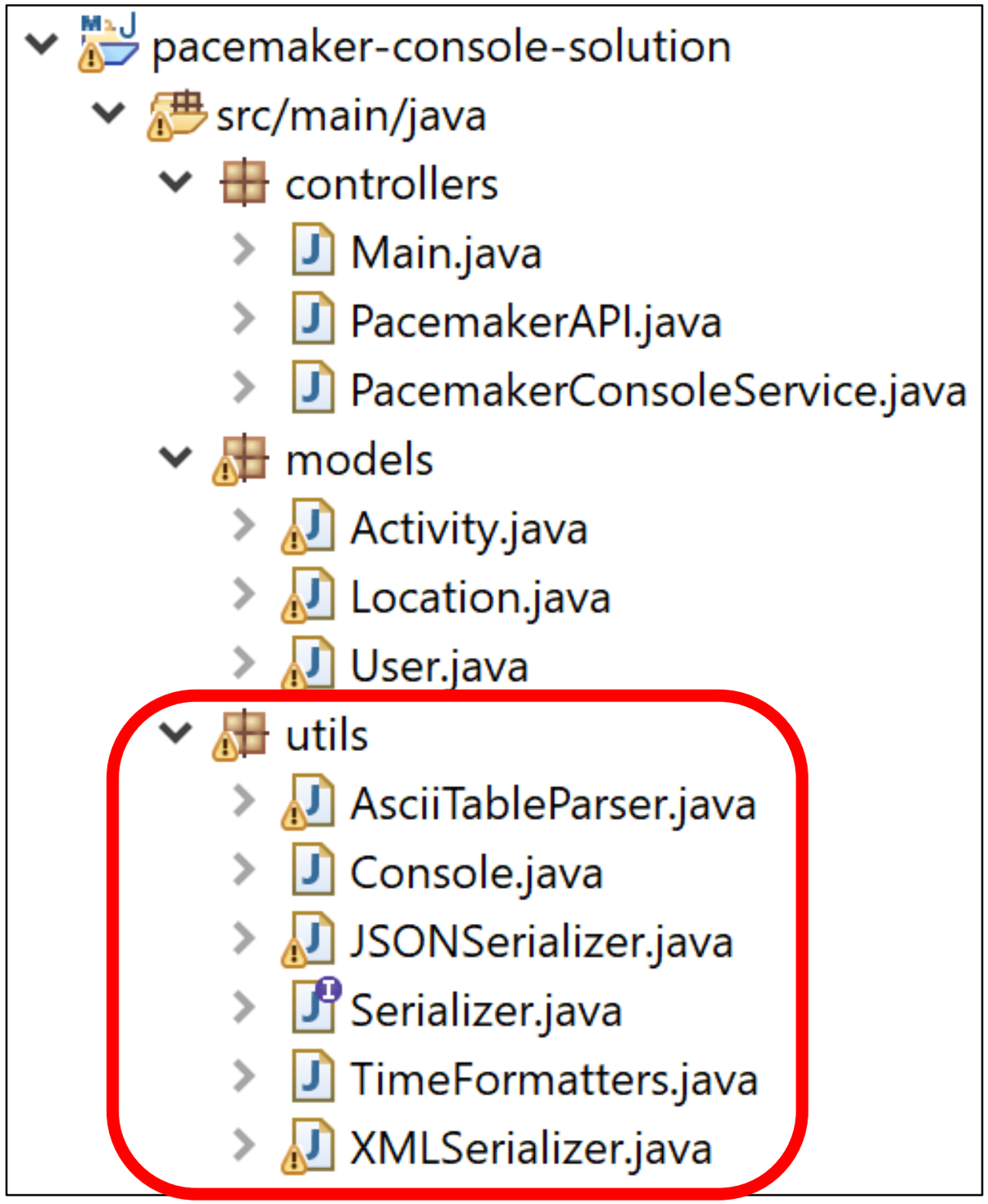
Activity and Location Models, with associated tests.



```java
public class UserTest {

  User homer = new User("homer", "simpson", "homer@simpson.com", "secret");

  @Test
  public void testCreate() {
    assertEquals("homer", homer.firstName);
    assertEquals("simpson", homer.lastName);
    assertEquals("homer@simpson.com", homer.email);
    assertEquals("secret", homer.password);
  }

  @Test
  public void testIds() {
    Set<String> ids = new HashSet<>();
    for (User user : users) {
      ids.add(user.id);
    }
    assertEquals(users.size(), ids.size());
  }

  @Test
  public void testToString() {
    assertEquals("User{" + homer.id + ", homer, simpson, secret, homer@simpson.com, {}}",
        homer.toString());
  }

  @Test
  public void tesAddActivity() {
    Activity activity = new Activity("walk", "fridge", 0.001, "11:2:2012 9:00:00", "20:00:00");
    homer.activities.put(activity.id,activity);
    System.out.println(homer);
    assertEquals("User{" + homer.id + ", homer, simpson, secret, homer@simpson.com, {"
                    + activity.id + "=Activity{" + activity.id + ", walk, fridge, 0.001, []}}}",
                    homer.toString());
  }
}
```

Refactoring UserTest to cater for activities.

Utilities


pacemaker-console-solution
src/main/java
controllers
Main.java
PacemakerAPI.java
PacemakerConsoleService.java
models
Activity.java
Location.java
User.java
utils
AsciiTableParser.java
Console.java
JSONSerializer.java
Serializer.java
TimeFormatters.java
XMLSerializer.java

code coverage (7a)

Run
→ Coverage...

Coverage Configurations

Create, manage, and run configurations

Coverage of a JUnit test run.

Java Application
JUnit
pacemaker-console-solution
UserTest

Filter matched 4 of 14 items

Name: pacemaker-console-solution

Test | Coverage | (x)= Arguments | Classpath | JRE | Source | Environment | »1

Analysis scope:

- pacemaker-console-solution - src/main/java
- pacemaker-console-solution - src/test/java
- pacemaker-console-solution - animal-sniffer-annotations-1.14.jar
- pacemaker-console-solution - error_prone_annotations-2.0.18.jar
- pacemaker-console-solution - guava-23.0.jar
- pacemaker-console-solution - hamcrest-core-1.3.jar
- pacemaker-console-solution - j2objc-annotations-1.1.jar
- pacemaker-console-solution - joda-time-2.9.9.jar
- pacemaker-console-solution - jsr305-1.3.9.jar
- pacemaker-console-solution - junit-4.12.jar

Select All | Deselect All

Revert | Apply

Coverage | Close

Package Explorer | JUnit

pacemaker-console-solution [Runner: JUnit 4]

Finished after 0.452 seconds

Runs: 10/10 Errors: 0 Failures: 0

- models.UserTest [Runner: JUnit 4] (0.340 s)
- models.ActivityTest [Runner: JUnit 4] (0.005 s)
- models.LocationTest [Runner: JUnit 4] (0.002 s)

Failure Trace

Activity.java

```
1  package models;
3  import static com.google.common.base.MoreObjects.toStringHelper;
16
17 public class Activity implements Serializable {
18
19   public String id;
20   public String type;
21   public String location;
22   public double distance;
23   public DateTime starttime;
24   public Duration duration;
25   public List<Location> route = new ArrayList<>();
26
27   public Activity() {
28   }
29
30   public Activity(String type, String location, double distance, String start, Stri
31     this.id = UUID.randomUUID().toString();
```

Problems | Javadoc | Declaration | Console | Coverage

pacemaker-console-solution (20-Oct-2017 12:18:21)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
pacemaker-console-solution	38.9 %	166	261	427
src/main/java	38.9 %	166	261	427
models	33.2 %	128	258	386
Activity.java	33.7 %	56	110	166
User.java	32.4 %	46	96	142
Location.java	33.3 %	26	52	78
utils	92.7 %	38	3	41
TimeFormatters.java	92.7 %	38	3	41

Generating a Coverage Report

- Manually: **File**, **Export...**, **Run/Debug**, **Coverage Session**.
- Via Maven to the target directory: using the plugin below and running the **mvn jacoco:report** command.

```
<plugin>
<groupId>org.jacoco</groupId>
<artifactId>jacoco-maven-plugin</artifactId>
<version>0.7.9</version>
    <executions>
        <execution>
            <id>default-prepare-agent</id>
            <goals>
                <goal>prepare-agent</goal>
            </goals>
        </execution>
        <execution>
            <id>default-report</id>
            <phase>prepare-package</phase>
            <goals>
                <goal>report</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

file:///C:/Users/Siobhan/Desktop/agile%20workspace/pacemaker-console-solution/target/site/jacoco/index.html

pacemaker-console-solution

pacemaker-console-solution

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
models		33%		0%	34	40	40	81	18	24	0	3
utils		92%		n/a	1	6	1	13	1	6	0	1
Total	261 of 427	38%	32 of 32	0%	35	46	41	94	19	30	0	4

file:///C:/Users/Siobhan/Desktop/agile%20workspace/pacemaker-console-solution/target/site/jacoco/models/index.html

pacemaker-console-solution > models

models

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
Activity		33%		0%	15	17	17	34	8	10	0	1
User		32%		0%	13	15	15	30	7	9	0	1
Location		33%		0%	6	8	8	17	3	5	0	1
Total	258 of 386	33%	32 of 32	0%	34	40	40	81	18	24	0	3

PacemakerAPI, partial implementation (7b)

PacemakerAPI.java

```java
public class PacemakerAPI {

  private Map<String, User> emailIndex = new HashMap<>();
  private Map<String, User> userIndex = new HashMap<>();
  private Map<String, Activity> activitiesIndex = new HashMap<>();

  public PacemakerAPI() {}

  public Collection<User> getUsers() {
    return userIndex.values();
  }

  public void deleteUsers() {
    userIndex.clear();
    emailIndex.clear();
  }

  public User createUser(String firstName, String lastName, String email, String password) {
    User user = new User(firstName, lastName, email, password);
    emailIndex.put(email, user);
    userIndex.put(user.id, user);
    return user;
  }

  public Activity createActivity(String id, String type, String location, double distance,
      String starttime, String duration) {
    Activity activity = null;
    Optional<User> user = Optional.fromNullable(userIndex.get(id));
    if (user.isPresent()) {
      activity = new Activity(type, location, distance, starttime, duration);
      user.get().activities.put(activity.id, activity);
      activitiesIndex.put(activity.id, activity);
    }
    return activity;
  }

  public void addLocation(String id, double latitude, double longitude) {
    Optional<Activity> activity = Optional.fromNullable(activitiesIndex.get(id));
    if (activity.isPresent()) {
      activity.get().route.add(new Location(latitude, longitude));
    }
  }

public List<Activity> listActivities(String userId, String sortBy) {
    return null;
  }

public Activity getActivity(String id) {
    return activitiesIndex.get(id);
  }

  public Collection<Activity> getActivities(String id) {
    Collection<Activity> activities = null;
    Optional<User> user = Optional.fromNullable(userIndex.get(id));
    if (user.isPresent()) {
      activities = user.get().activities.values();
    }
    return activities;
  }


  public User getUserByEmail(String email) {
    return emailIndex.get(email);
  }

  public User getUser(String id) {
    return userIndex.get(id);
  }

  public User deleteUser(String id) {
    User user = userIndex.remove(id);
    return emailIndex.remove(user.email);
  }
}
```

PacemakerAPITest.java (sample of tests)

```java
@Test
public void testEquals() {
  User homer = new User("homer", "simpson", "homer@simpson.com", "secret");
  User homer2 = new User("homer", "simpson", "homer@simpson.com", "secret");
  User bart = new User("bart", "simpson", "bartr@simpson.com", "secret");

  assertEquals(homer, homer);
  assertEquals(homer, homer2);
  assertNotEquals(homer, bart);

  assertSame(homer, homer);
  assertNotSame(homer, homer2);
}
```

```java
@Test
 public void testDeleteUsers() {
   assertEquals(users.size(), pacemaker.getUsers().size());
   User marge = pacemaker.getUserByEmail("marge@simpson.com");
   pacemaker.deleteUser(marge.id);
   assertEquals(users.size() - 1, pacemaker.getUsers().size());
 }
```

```java
@Test
public void testAddActivityWithMultipleLocation() {
  User marge = pacemaker.getUserByEmail("marge@simpson.com");
  Activity testActivity = margeActivities.get(0);
  String activityId = pacemaker
      .createActivity(marge.id, testActivity.type, testActivity.location, testActivity.distance,
          parseDateTime(testActivity.starttime), parseDuration(testActivity.duration)).id;

  locations.forEach(
      location -> pacemaker.addLocation(activityId, location.latitude, location.longitude));

  Activity activity = pacemaker.getActivity(activityId);
  assertEquals(locations.size(), activity.route.size());

  int i = 0;
  for (Location location : activity.route) {
    assertEquals(location, locations.get(i));
    i++;
  }
}
```

Serialization (7b)

Briefly do XMLSerializer and JSON serializer (from solution) and the testing approach.

Lambdas…sorting….

pacemakerAPI – focus on the listActivities method and the lambdas / comparators…

PacemakerAPI.java

```
public List<Activity> listActivities(String userId, String sortBy) {
   List<Activity> activities = new ArrayList<>();
   activities.addAll(userIndex.get(userId).activities.values());
   switch (sortBy) {
     case "type":
       activities.sort((a1, a2) -> a1.type.compareTo(a2.type));
       break;
     case "location":
       activities.sort((a1, a2) -> a1.location.compareTo(a2.location));
       break;
     case "distance":
       activities.sort((a1, a2) -> Double.compare(a1.distance, a2.distance));
       break;
     case "date":
       activities
           .sort((a1, a2) -> DateTimeComparator.getInstance().compare(a1.starttime, a2.starttime));
       break;
     case "duration":
       activities
           .sort((a1, a2) -> {
             if (a1.duration.getStandardSeconds() > a2.duration.getStandardSeconds()) {
               return 1;
             } else {
               return -1;
             }
           });
       break;
   }
   return activities;
}
```

PacemakerAPITest.java

```
@Test
 public void testSortedReports() {
   User homer = pacemaker.createUser("homer", "simpson", "homer@simpson.com", "secret");
   activities.forEach(
       activity -> pacemaker
           .createActivity(homer.id, activity.type, activity.location, activity.distance,
               parseDateTime(activity.starttime), parseDuration(activity.duration))
   );
   List<Activity> activities = pacemaker.listActivities(homer.id, "type");

   for (int i = 0; i < activities.size() - 1; i++) {
     assertTrue(activities.get(i).type.compareTo(activities.get(i + 1).type) <= 0);
   }
   activities = pacemaker.listActivities(homer.id, "location");
   for (int i = 0; i < activities.size() - 1; i++) {
     assertTrue(activities.get(i).location.compareTo(activities.get(i + 1).location) <= 0);
   }
   activities = pacemaker.listActivities(homer.id, "distance");
   for (int i = 0; i < activities.size() - 1; i++) {
     assertTrue(activities.get(i).distance <= activities.get(i + 1).distance);
   }
   activities = pacemaker.listActivities(homer.id, "date");
   for (int i = 0; i < activities.size() - 1; i++) {
     assertTrue( DateTimeComparator.getInstance().compare(activities.get(i).starttime,
activities.get(i+1).starttime) <= 0);
   }
   activities = pacemaker.listActivities(homer.id, "duration");
   for (int i = 0; i < activities.size() - 1; i++) {
     assertTrue( activities.get(i).duration.getStandardSeconds() <
activities.get(i+1).duration.getStandardSeconds());
   }
 }
```

UX (7b)

Console, parser, pacemaker console service