Algorithms

Produced by

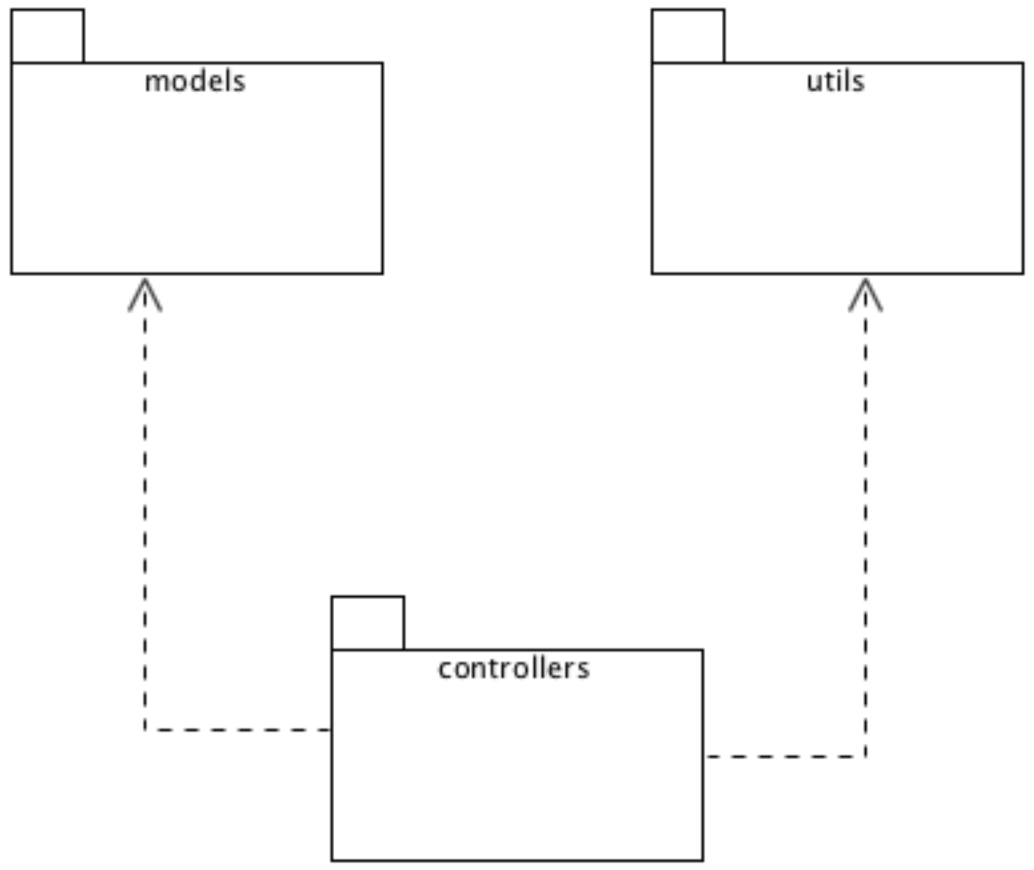
Eamonn de Leastar (edeleastar@wit.ie)

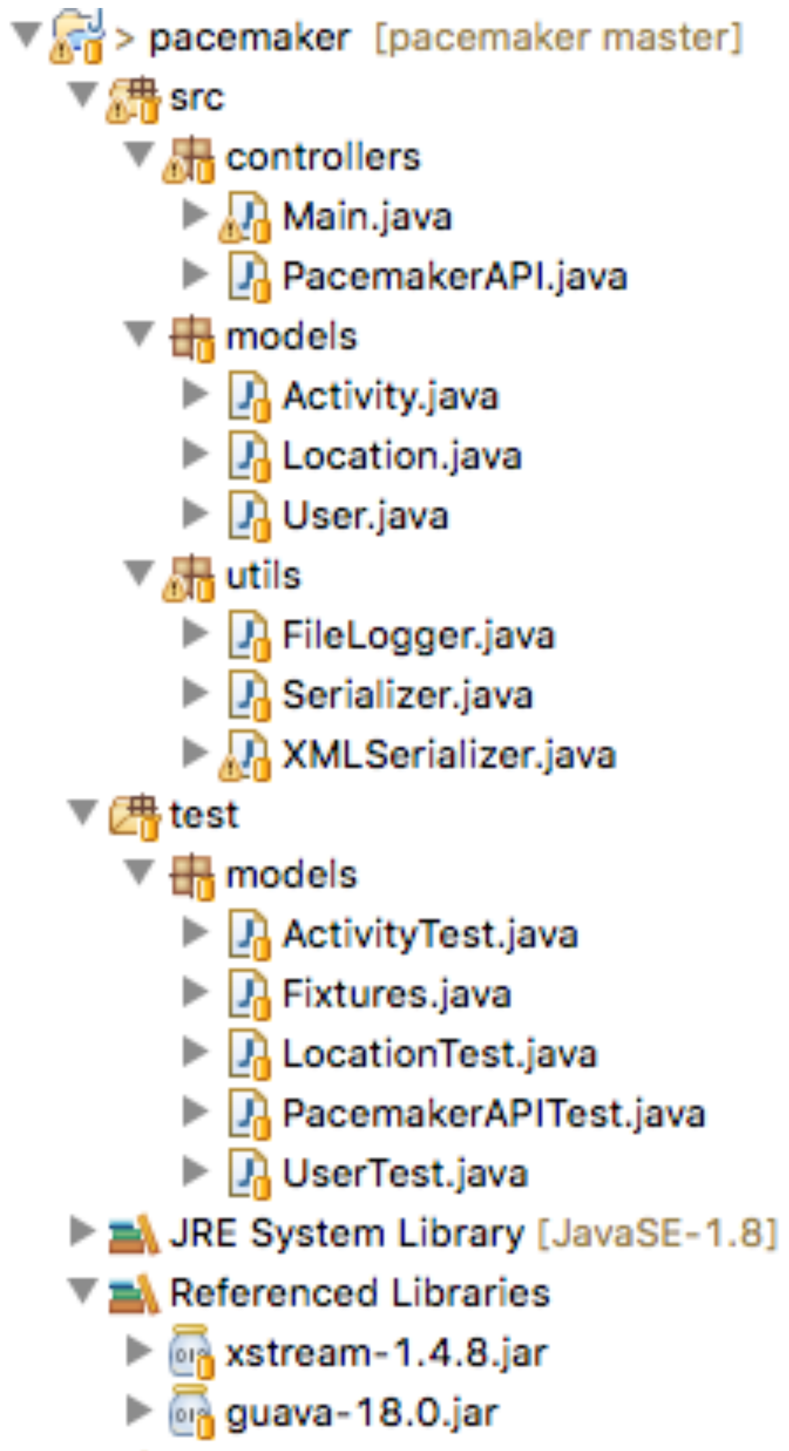
Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie

pacemaker model





```
▼ > pacemaker [pacemaker master]
  ▼ src
    ▼ controllers
      ▶ Main.java
      ▶ PacemakerAPI.java
    ▼ models
      ▶ Activity.java
      ▶ Location.java
      ▶ User.java
    ▼ utils
      ▶ FileLogger.java
      ▶ Serializer.java
      ▶ XMLSerializer.java
  ▼ test
    ▼ models
      ▶ ActivityTest.java
      ▶ Fixtures.java
      ▶ LocationTest.java
      ▶ PacemakerAPITest.java
      ▶ UserTest.java
  ▶ JRE System Library [JavaSE-1.8]
  ▼ Referenced Libraries
    ▶ xstream-1.4.8.jar
    ▶ guava-18.0.jar
```

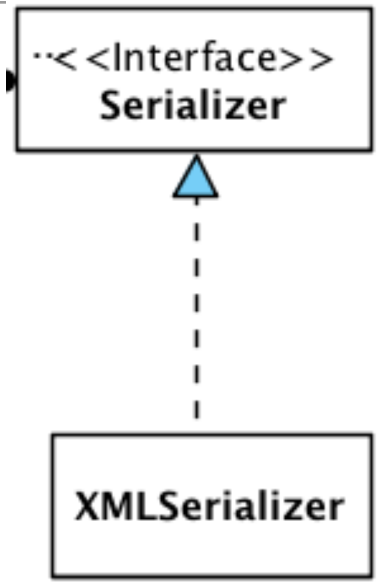
Serialiser

- An abstraction to encapsulate persistence mechanism
- Push objects onto the stack
- All objects pushed are then saved in a single ‘write’ operation
- If read is called, a persistence state is restored… and can be recovered by popping the stack

```java
public interface Serializer
{
  void push(Object o);
  Object pop();
  void write() throws Exception;
  void read() throws Exception;
}
```

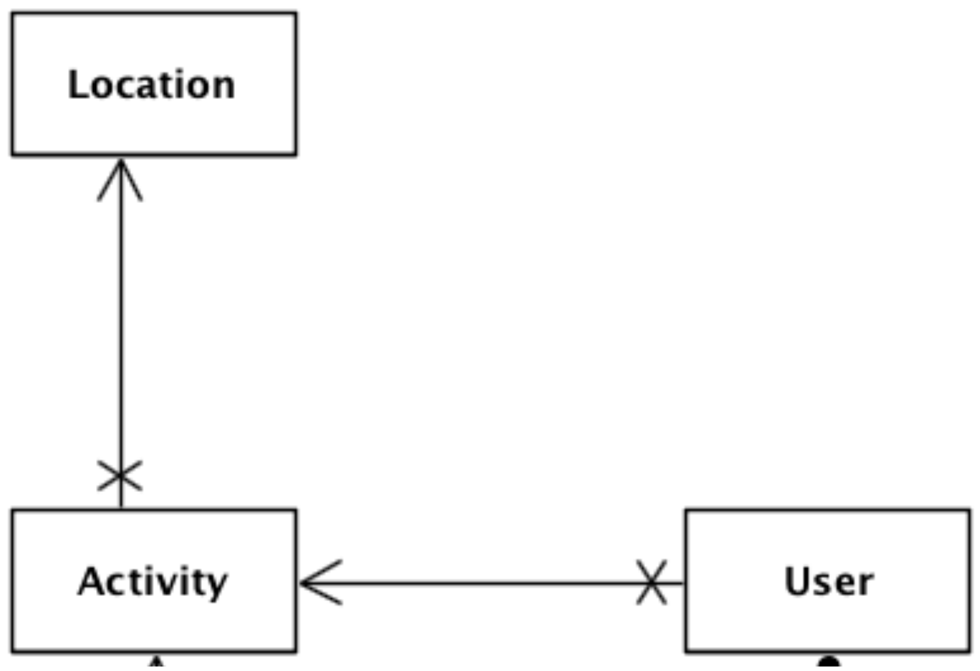
XML Serliaizer

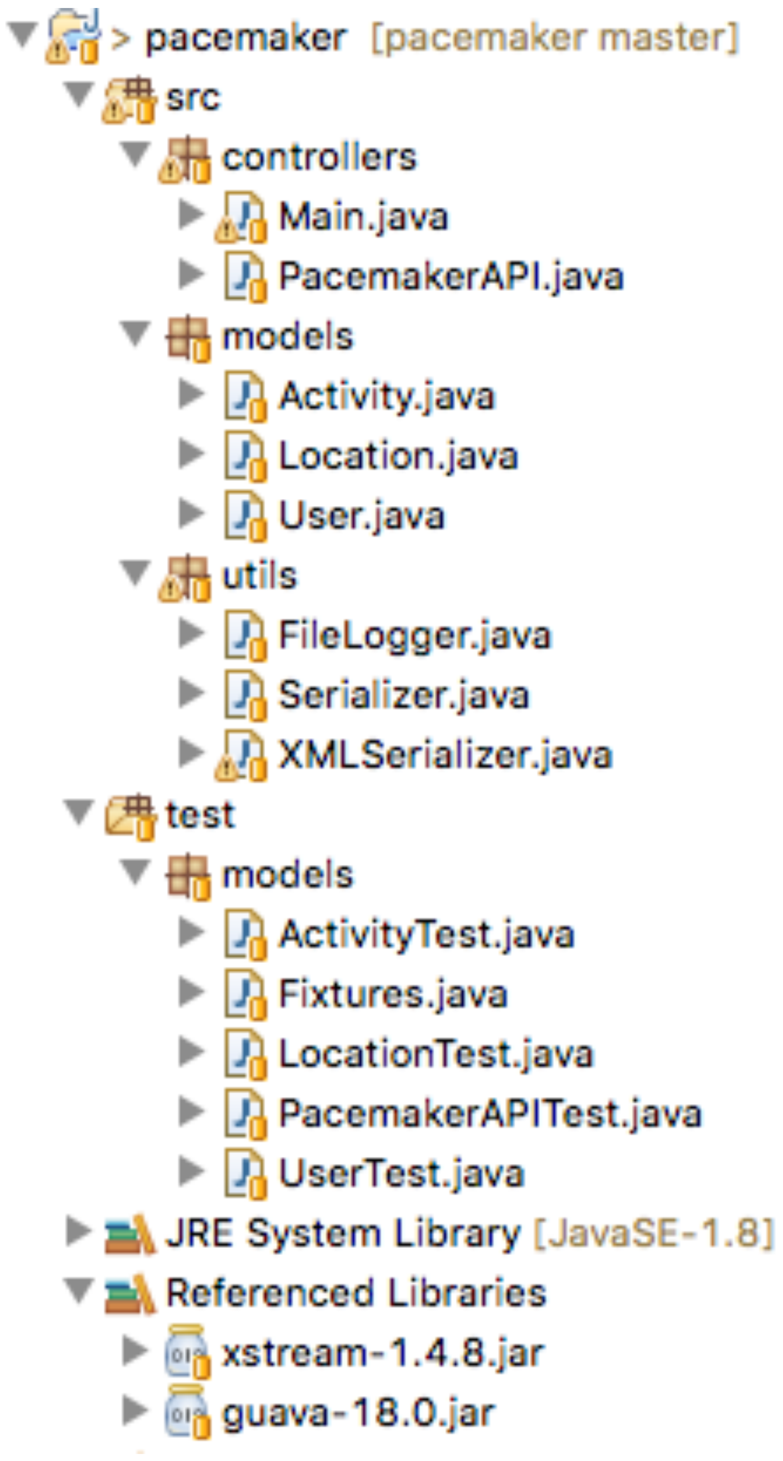
```
public class XMLSerializer implements Serializer
{
  private Stack stack = new Stack();
  private File file;

  public XMLSerializer(File file)
  {
    this.file = file;
  }

  public void push(Object o)
  {
    stack.push(o);
  }

  public Object pop()
  {
    return stack.pop();
  }

  @SuppressWarnings("unchecked")
  public void read() throws Exception
  {
    ObjectInputStream is = null;

    try
    {
      XStream xstream = new XStream(new DomDriver());
      is = xstream.createObjectInputStream(new FileReader
      stack = (Stack) is.readObject();
    }
    finally
    {
      if (is != null)
      {
        is.close();
      }
    }
  }
```



```
  public void write() throws Exception
  {
    ObjectOutputStream os = null;

    try
    {
      XStream xstream = new XStream(new DomDriver());
      os = xstream.createObjectOutputStream(new FileWriter(file));
      os.writeObject(stack);
    }
    finally
    {
      if (os != null)
      {
        os.close();
      }
    }
  }
}
```

Models




```
▼ > pacemaker [pacemaker master]
  ▼ src
    ▼ controllers
      ▶ Main.java
      ▶ PacemakerAPI.java
    ▼ models
      ▶ Activity.java
      ▶ Location.java
      ▶ User.java
    ▼ utils
      ▶ FileLogger.java
      ▶ Serializer.java
      ▶ XMLSerializer.java
  ▼ test
    ▼ models
      ▶ ActivityTest.java
      ▶ Fixtures.java
      ▶ LocationTest.java
      ▶ PacemakerAPITest.java
      ▶ UserTest.java
  ▶ JRE System Library [JavaSE-1.8]
  ▼ Referenced Libraries
    ▶ xstream-1.4.8.jar
    ▶ guava-18.0.jar
```

pacemaker model

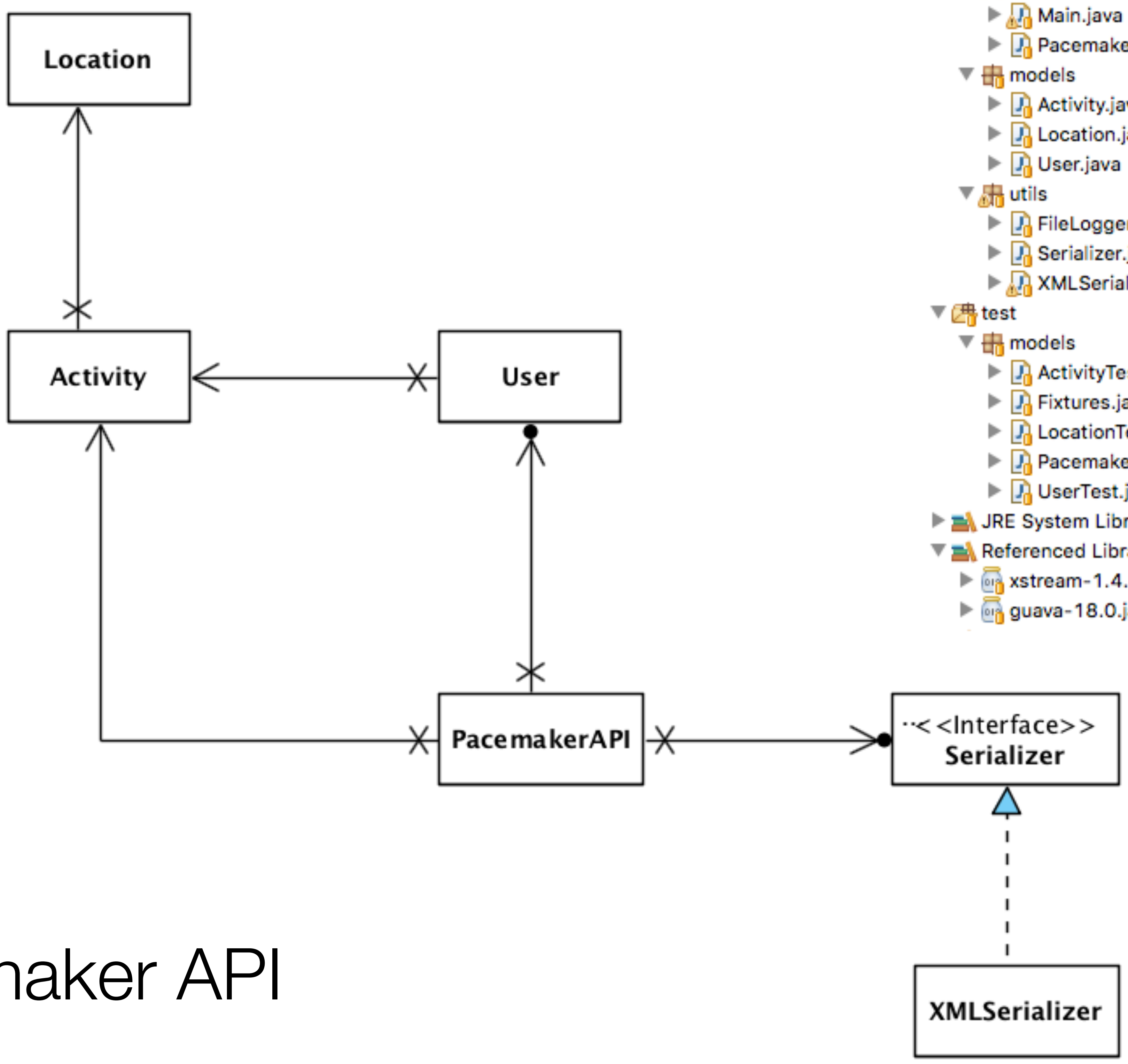
```java
public class User
{
  static Long   counter = 0l;

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

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

  //...
}
```

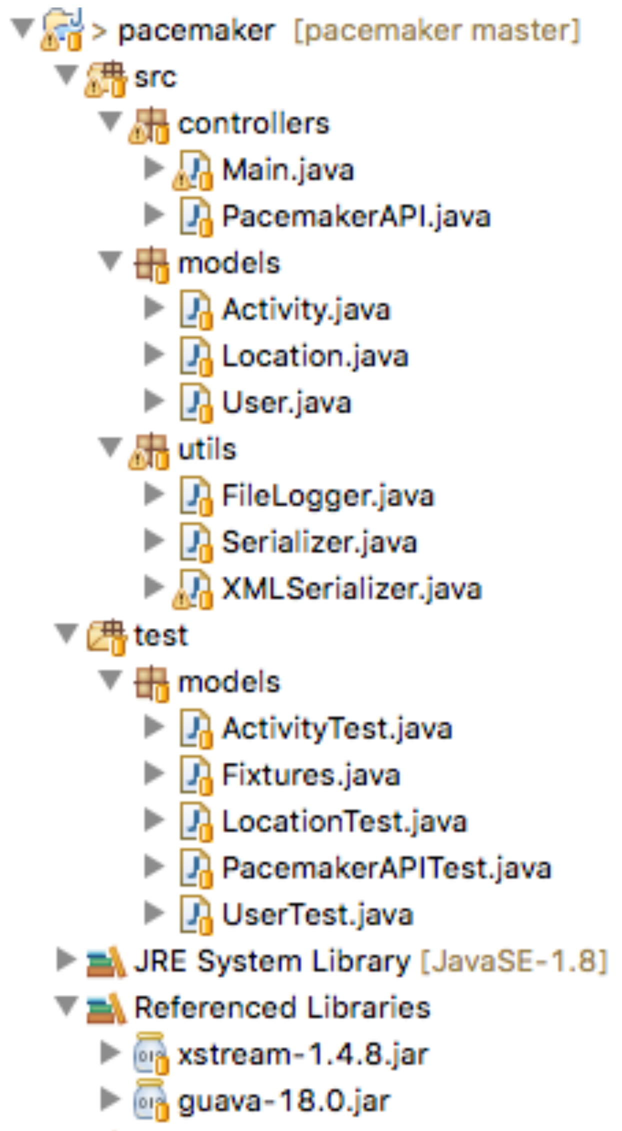
```java
public class Activity
{
  static Long   counter = 0l;

  public Long   id;
  public String type;
  public String location;
  public double distance;

  public List<Location> route = new ArrayList<>();

  //...
}
```

```java
public class Location
{
  static Long   counter = 0l;

  public Long  id;
  public float latitude;
  public float longitude;

  //...
}
```

pacemaker model - equals/toString/hashCode

```
public class User
{
  //...
  @Override
  public String toString()
  {
    return toStringHelper(this).addValue(id)
                               .addValue(firstName)
                               .addValue(lastName)
                               .addValue(password)
                               .addValue(email)
                               .addValue(activities)
                               .toString();
  }
  @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 int hashCode()
  {
     return Objects.hashCode(this.id, this.lastName, this.firstName, this.email, this.password);
  }
}
```




```
pacemaker [pacemaker master]
  src
    controllers
      Main.java
      PacemakerAPI.java
    models
      Activity.java
      Location.java
      User.java
    utils
      FileLogger.java
      Serializer.java
      XMLSerializer.java
  test
    models
      ActivityTest.java
      Fixtures.java
      LocationTest.java
      PacemakerAPITest.java
      UserTest.java
  JRE System Library [JavaSE-1.8]
  Referenced Libraries
    xstream-1.4.8.jar
    guava-18.0.jar
```

Pacemaker API

PacemakerAPI (1)

- Implement the core features of the pacemaker service
- Not concerned with UI at this stage

```
public class PacemakerAPI
{
  private Map<Long,   User>   userIndex       = new HashMap<>();
  private Map<String, User>   emailIndex      = new HashMap<>();
  private Map<Long, Activity> activitiesIndex = new HashMap<>();

  //...

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

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

  public void deleteUser(Long id)
  {
    User user = userIndex.remove(id);
    emailIndex.remove(user.email);
  }

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

PacemakerAPI (2)

```java
public class PacemakerAPI
{
  private Map<Long,   User>    userIndex       = new HashMap<>();
  private Map<String, User>    emailIndex      = new HashMap<>();
  private Map<Long, Activity> activitiesIndex = new HashMap<>();

  //...


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

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

pacemaker persistence

```
public interface Serializer
{
  void push(Object o);
  Object pop();
  void write() throws Exception;
  void read() throws Exception;
}
```

```
public class PacemakerAPI
{
  private Map<Long,   User>    userIndex       = new HashMap<>();
  private Map<String, User>    emailIndex      = new HashMap<>();
  private Map<Long, Activity> activitiesIndex = new HashMap<>();

  private Serializer serializer;

  public PacemakerAPI(Serializer serializer)
  {
    this.serializer = serializer;
  }

  @SuppressWarnings("unchecked")
  public void load() throws Exception
  {
    serializer.read();
    activitiesIndex = (Map<Long, Activity>) serializer.pop();
    emailIndex      = (Map<String, User>)   serializer.pop();
    userIndex       = (Map<Long, User>)     serializer.pop();
  }

  public void store() throws Exception
  {
    serializer.push(userIndex);
    serializer.push(emailIndex);
    serializer.push(activitiesIndex);
    serializer.write();
  }
```

Persistence Tests

PersistenceTest - fixtures

```
public class Fixtures
{
  public static User[] users =
  {
    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")
  };

  public static Activity[] activities =
  {
    new Activity ("walk",  "fridge", 0.001),
    new Activity ("walk",  "bar",    1.0),
    new Activity ("run",   "work",   2.2),
    new Activity ("walk",  "shop",   2.5),
    new Activity ("cycle", "school", 4.5)
  };

  public static Location[] locations =
  {
    new Location(23.3f, 33.3f),
    new Location(34.4f, 45.2f),
    new Location(25.3f, 34.3f),
    new Location(44.4f, 23.3f)
  };
}
```

```
public class PersistenceTest
{
  PacemakerAPI pacemaker;

  void populate (PacemakerAPI pacemaker)
  {
    for (User user : users)
    {
      pacemaker.createUser(user.firstName, user.lastName, user.email, user.p
    }

    User user1 = pacemaker.getUserByEmail(users[0].email);
    Activity activity = pacemaker.createActivity(user1.id, activities[0].typ
    pacemaker.createActivity(user1.id, activities[1].type, activities[1].location, activities[1].distance);
    User user2 = pacemaker.getUserByEmail(users[1].email);
    pacemaker.createActivity(user2.id, activities[2].type, activities[2].location, activities[2].distance);
    pacemaker.createActivity(user2.id, activities[3].type, activities[3].location, activities[3].distance);

    for (Location location : locations)
    {
      pacemaker.addLocation(activity.id, location.latitude, location.longitude);
    }
  }

  void deleteFile(String fileName)
  {
    File datastore = new File ("testdatastore.xml");
    if (datastore.exists())
    {
      datastore.delete();
    }
  }
```

Verify Fixture

```java
@Test
public void testPopulate()
{
  pacemaker = new PacemakerAPI(null);
  assertEquals(0, pacemaker.getUsers().size());
  populate (pacemaker);

  assertEquals(users.length, pacemaker.getUsers().size());
  assertEquals(2, pacemaker.getUserByEmail(users[0].email).activities.size());
  assertEquals(2, pacemaker.getUserByEmail(users[1].email).activities.size());
  Long activityID = pacemaker.getUserByEmail(users[0].email).activities.keySet().iterator().next();
  assertEquals(locations.length, pacemaker.getActivity(activityID).route.size());
}
```

Serializer Test (XML)

```
@Test
public void testXMLSerializer() throws Exception
{
  String datastoreFile = "testdatastore.xml";
  deleteFile (datastoreFile);

  Serializer serializer = new XMLSerializer(new File (datastoreFile));

  pacemaker = new PacemakerAPI(serializer);
  populate(pacemaker);
  pacemaker.store();

  PacemakerAPI pacemaker2 =  new PacemakerAPI(serializer);
  pacemaker2.load();

  assertEquals (pacemaker.getUsers().size(), pacemaker2.getUsers().size());
  for (User user : pacemaker.getUsers())
  {
    assertTrue (pacemaker2.getUsers().contains(user));
  }
  deleteFile ("testdatastore.xml");
}
```

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit