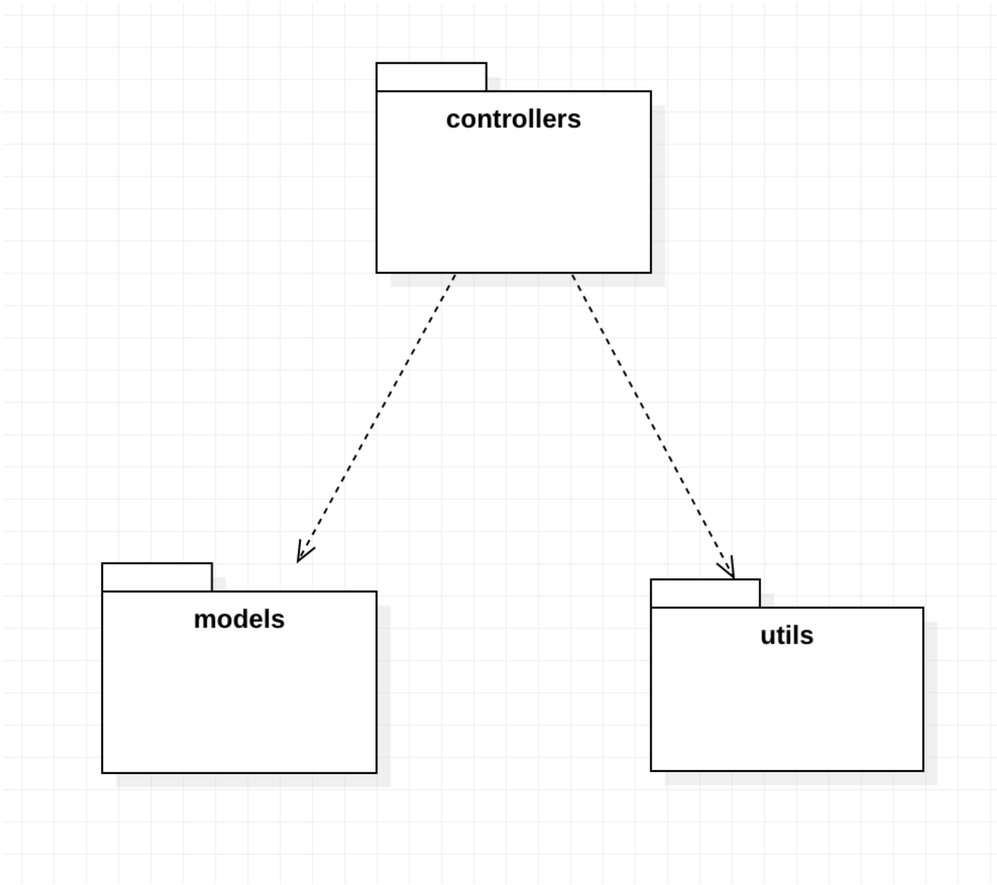
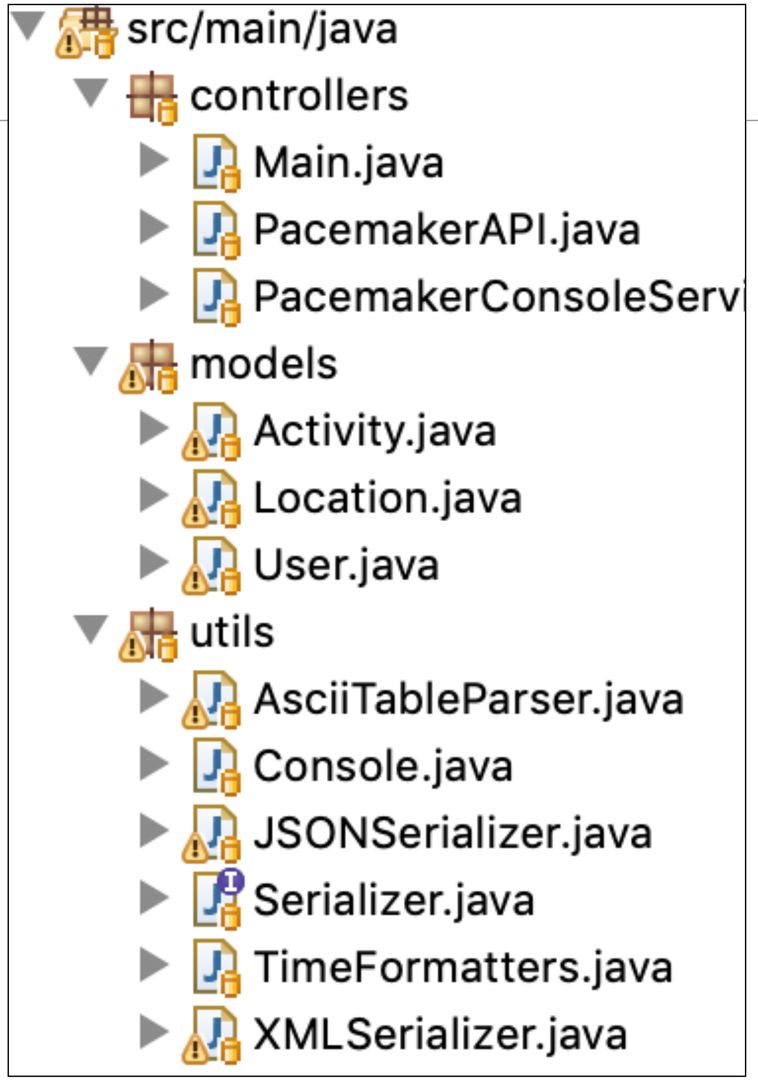
Assignment Solution

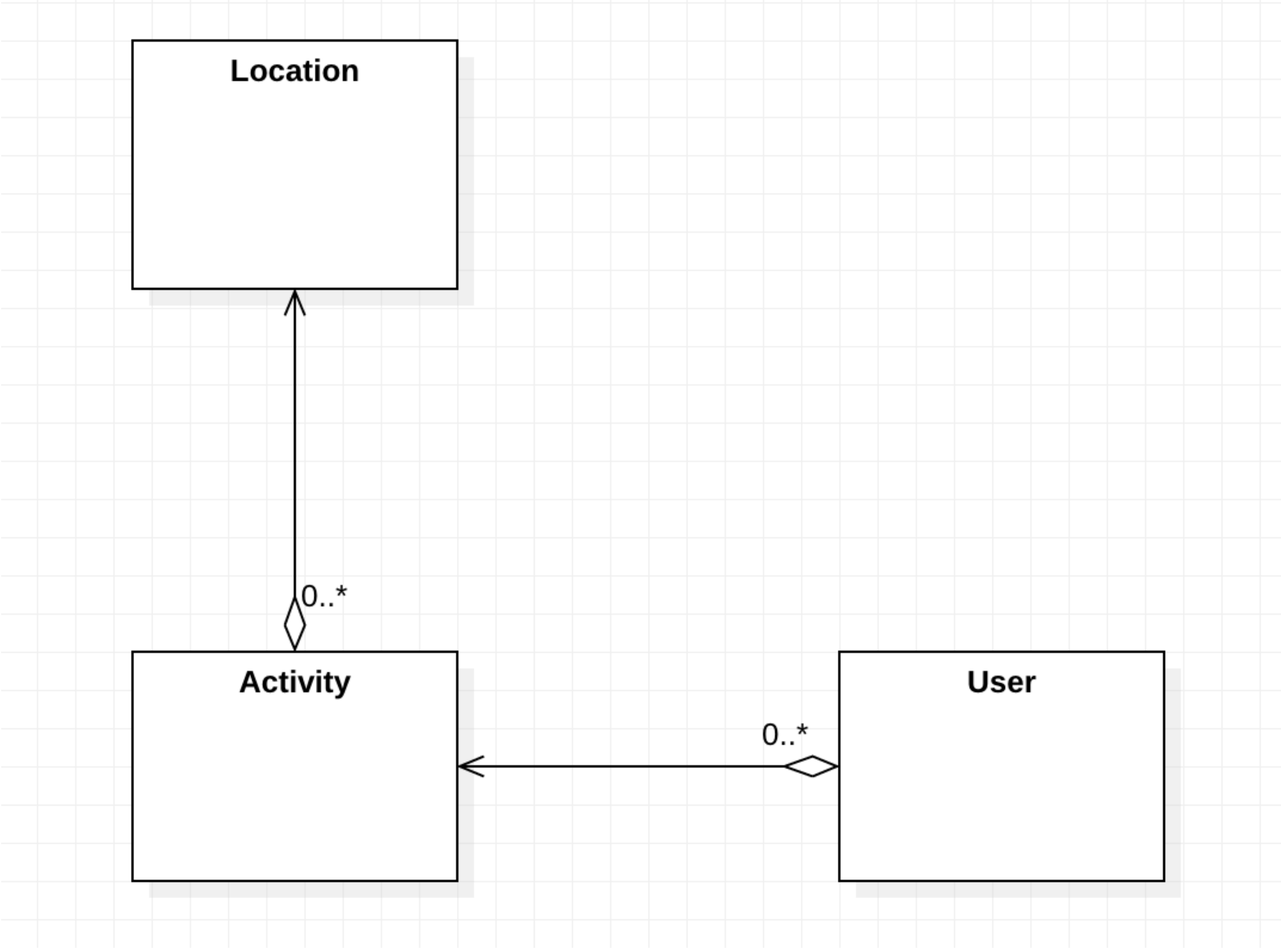
UML + Code

pacemaker-console-solution

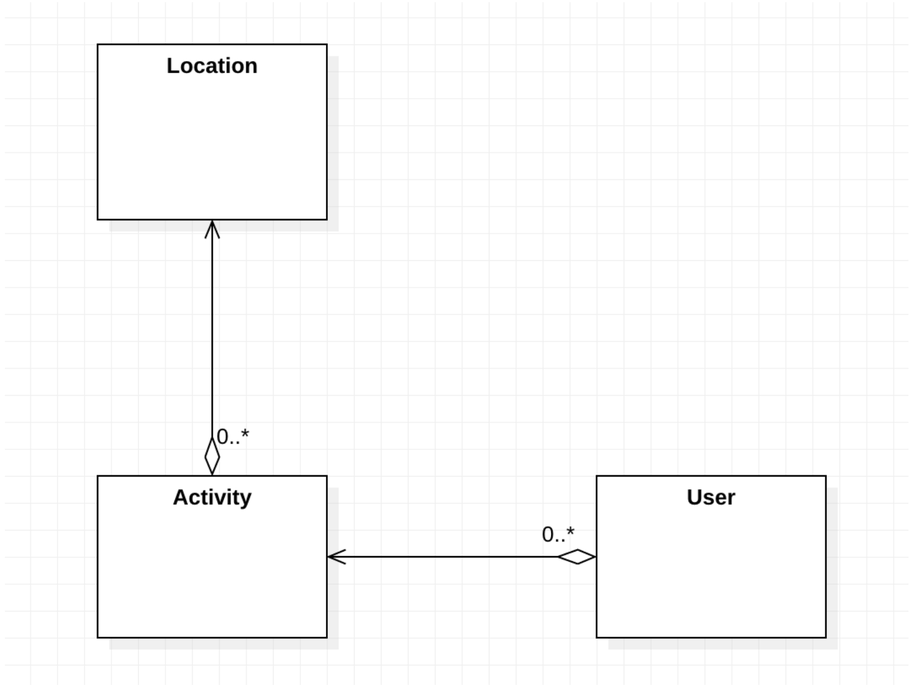




models

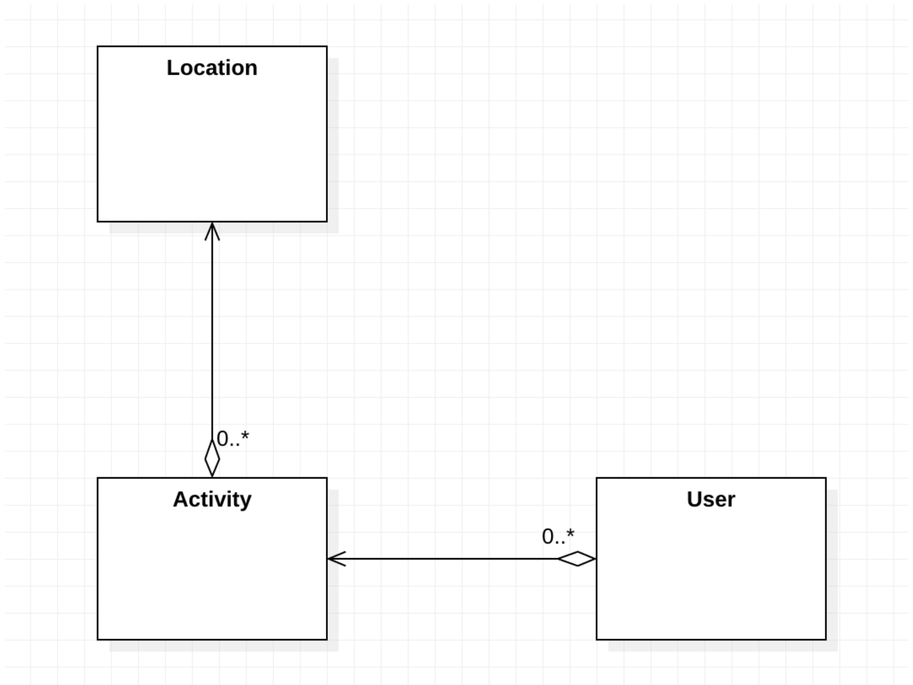


models - User



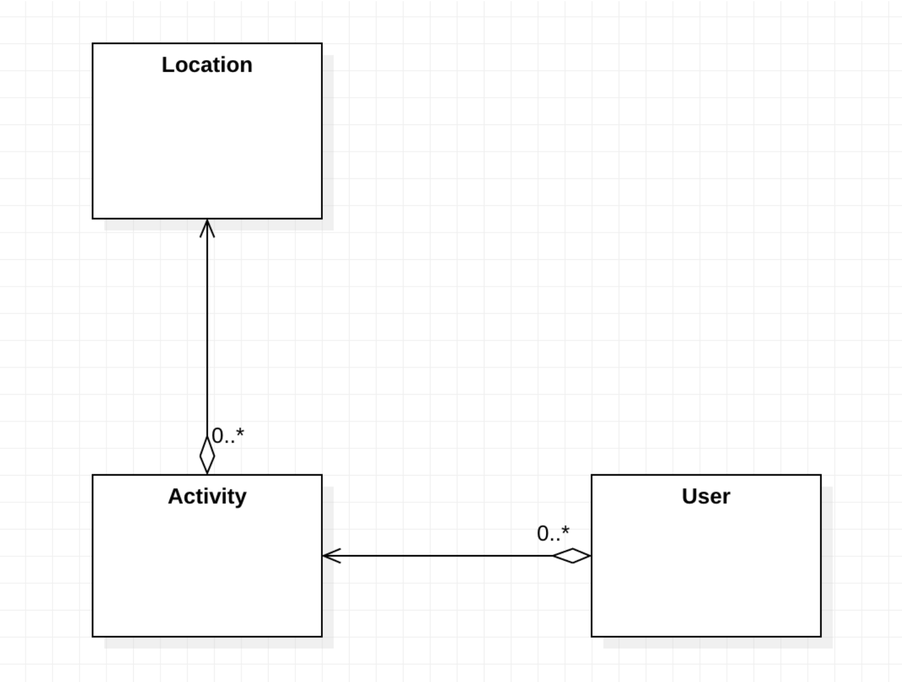

```java
public class User implements Serializable {

  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;
  }

  public User(String firstName, String lastName, String email, String password) {
    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);
  }
}
```

models - Activity



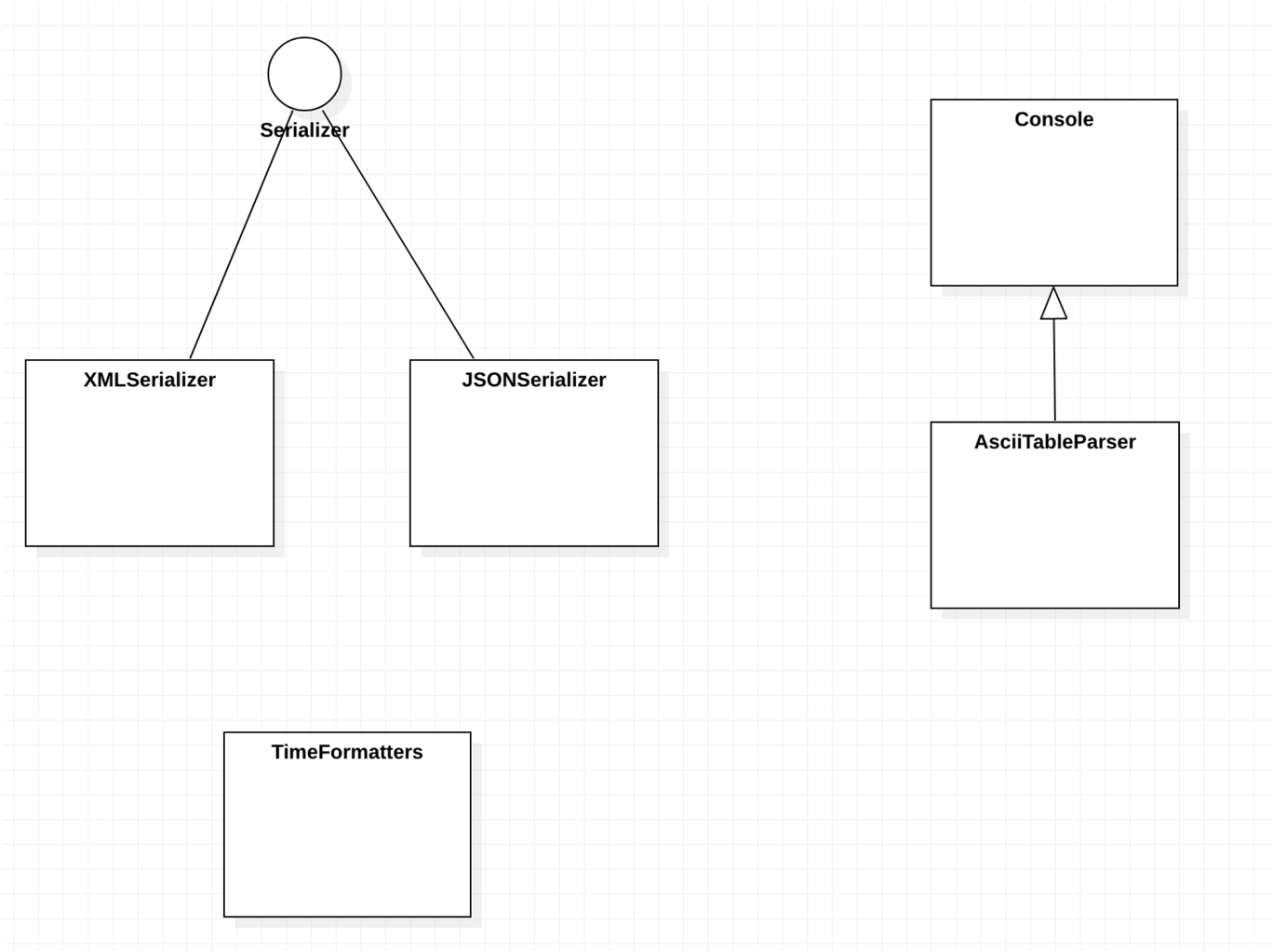

```java
public class Activity implements Serializable {

  public String id;
  public String type;
  public String location;
  public double distance;
  public DateTime starttime;
  public Duration duration;
  public List<Location> route = new ArrayList<>();

  public Activity() {
  }

  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);
  }

  public String getId() {
    return id;
  }

  public String getType() {
    return type;
  }

  public String getLocation() {
    return location;
  }

  public String getDistance() {
    return Double.toString(distance);
  }

  public String getRoute() {
    return route.toString();
  }

  @Override
  public boolean equals(final Object obj) {
    if (obj instanceof Activity) {
      final Activity other = (Activity) obj;
      return Objects.equal(type, other.type)
          && Objects.equal(location, other.location)
          && Objects.equal(distance, other.distance)
          && Objects.equal(starttime, other.starttime)
          && Objects.equal(duration, other.duration)
          && Objects.equal(route, other.route);
    } else {
      return false;
    }
  }

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

  @Override
  public int hashCode() {
    return Objects.hashCode(this.id, this.type, this.location, this.distance, this.starttime, this.duration);
  }
}
```

models - Location




```java
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);
  }
}
```

utils


Serializer
XMLSerializer
JSONSerializer
Console
AsciiTableParser
TimeFormatters

utils - Serializer

```
public interface Serializer {

  void push(Object o);

  Object pop();

  void write() throws Exception;

  void read() throws Exception;
}
```

utils - JSONSerilizer

```
public class JSONSerializer implements Serializer {

  private Stack stack = new Stack();
  private File file;

  public JSONSerializer(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 JettisonMappedXmlDriver());
      is = xstream.createObjectInputStream(new FileReader(file));
      stack = (Stack) is.readObject();
    } finally {
      if (is != null) {
        is.close();
      }
    }
  }

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

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

utils - XMLSerilizer

```
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(file));
      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();
      }
    }
  }
}
```

utils - TimeFormatters

```
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());
  }
}
```

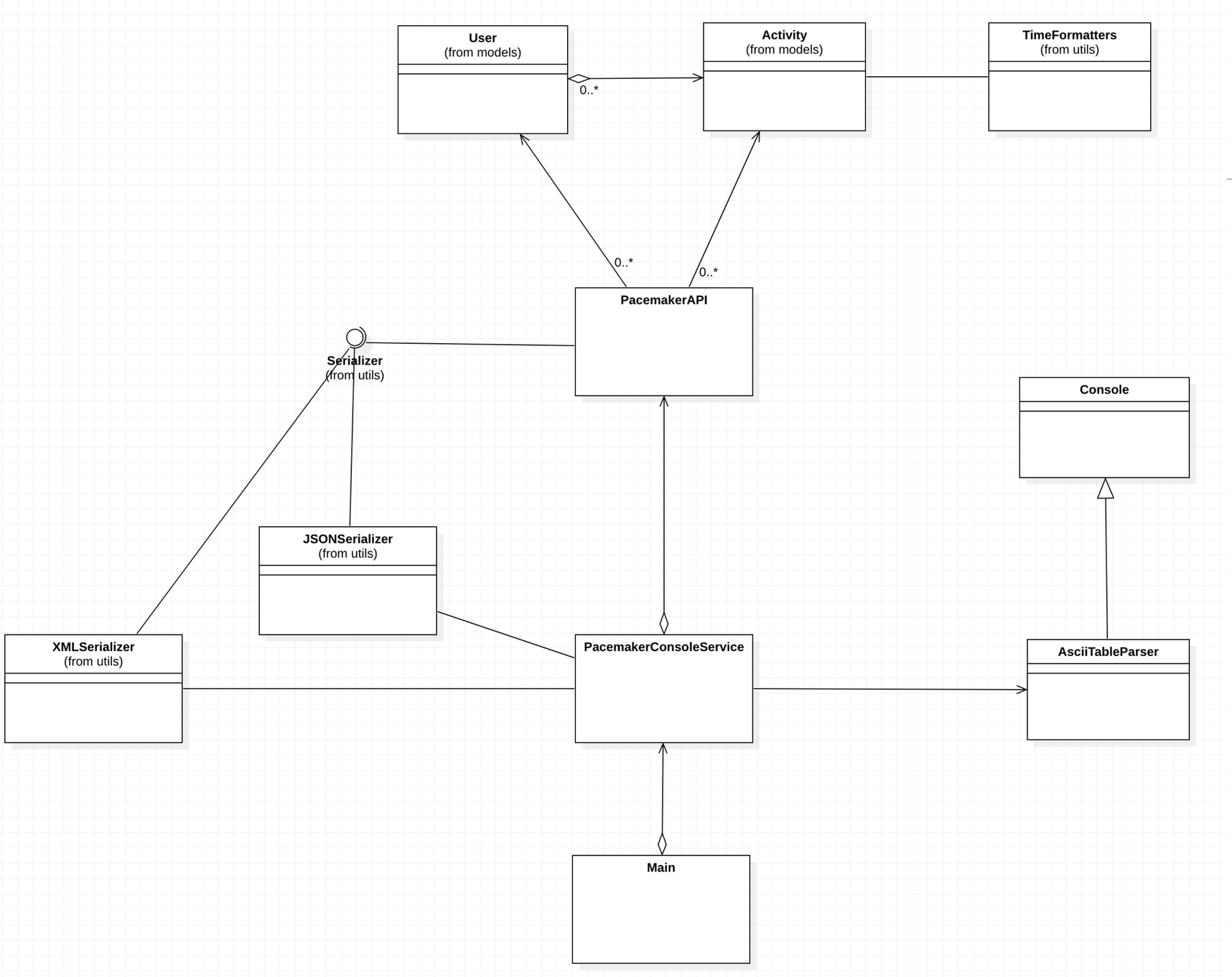
utils - Console

```
public class Console {

  public void println(String s) {
    System.out.println(s);
  }
  public void renderUser(User user) {
    System.out.println(user.toString());
  }

  public void renderUsers(Collection<User> users) {
    System.out.println(users.toString());
  }

  public void renderActivity(Activity activities) {
    System.out.println(activities.toString());
  }

  public void renderActivities(Collection<Activity> activities) {
    System.out.println(activities.toString());
  }
}
```

utils - AsciiTableParser

```java
public class AsciiTableParser extends Console {

  public void renderUser(User user) {
    if (user != null) {
      renderUsers(Arrays.asList(user));
      System.out.println("ok");
    } else {
      System.out.println("not found");
    }
  }

  public void renderUsers(Collection<User> users) {
    if (users != null) {
      if (!users.isEmpty()) {
        List<User> userList = new ArrayList<User>(users);
        IASCIITableAware asciiTableAware = new CollectionASCIITableAware<User>(userList, "id",
            "firstname",
            "lastname", "email");
        System.out.println(ASCIITable.getInstance().getTable(asciiTableAware));
      }
      System.out.println("ok");
    } else {
      System.out.println("not found");
    }
  }

  public void renderActivity(Activity activity) {
    if (activity != null) {
      renderActivities(Arrays.asList(activity));
      System.out.println("ok");
    } else {
      System.out.println("not found");
    }
  }

  public void renderActivities(Collection<Activity> activities) {
    if (activities != null) {
      if (!activities.isEmpty()) {
        List<Activity> activityList = new ArrayList(activities);
        IASCIITableAware asciiTableAware = new CollectionASCIITableAware<Activity>(activityList,
            "id",
            "type", "location", "distance", "starttime", "duration", "route");
        System.out.println(ASCIITable.getInstance().getTable(asciiTableAware));
      }
      System.out.println("ok");
    } else {
      System.out.println("not found");
    }
  }
}
```

controllers



controllers - PacemakerAPI

```
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 Serializer serializer;

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

  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 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;
  }
```

controllers - PacemakerAPI

```
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;
}

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));
  }
}
```

controllers - PacemakerAPI

```
  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);
  }

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

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

controllers - PacemakerConsoleService

```
public class PacemakerConsoleService {

  PacemakerAPI paceApi;
  File datastore = new File("datastore");
  Serializer xmlSerializer = new XMLSerializer(datastore);
  Serializer jsonSerializer = new JSONSerializer(datastore);
  Console console = new AsciiTableParser();

  public PacemakerConsoleService() throws Exception {
    paceApi = new PacemakerAPI(xmlSerializer);
    if (datastore.isFile()) {
      paceApi.load();
    }
  }

  @Command(description = "Create a new User")
  public void createUser(@Param(name = "first name") String firstName,
      @Param(name = "last name") String lastName,
      @Param(name = "email") String email, @Param(name = "password") String password) {
    console.renderUser(paceApi.createUser(firstName, lastName, email, password));
  }

  @Command(description = "Get a Users details")
  public void getUser(@Param(name = "email") String email) {
    console.renderUser(paceApi.getUserByEmail(email));
  }

  @Command(description = "Get all users details")
  public void getUsers() {
    console.renderUsers(paceApi.getUsers());
  }

  @Command(description = "Delete a User")
  public void deleteUser(@Param(name = "email") String email) {
    console.renderUser(paceApi.getUserByEmail(email));
  }
```

controllers - PacemakerConsoleService

```
@Command(description = "Add an activity")
public void addActivity(@Param(name = "user-id") String id,
    @Param(name = "type") String type,
    @Param(name = "location") String location,
    @Param(name = "distance") double distance,
    @Param(name = "starttime") String starttime,
    @Param(name = "duration") String duration) {
  console
      .renderActivity(paceApi.createActivity(id, type, location, distance, starttime, duration));
}

@Command(description = "Add a location to an activity")
public void addLocation(@Param(name = "activity-id") String id,
    @Param(name = "longitude") double longitude,
    @Param(name = "latitude") double latitude) {
  Optional<Activity> activity = Optional.fromNullable(paceApi.getActivity(id));
  if (activity.isPresent()) {
    paceApi.addLocation(activity.get().id, latitude, longitude);
    console.println("ok");
  } else {
    console.println("not found");
  }
}

@Command(description = "List a users activities")
public void listActivities(@Param(name = "user id") String id) {
  Optional<User> user = Optional.fromNullable(paceApi.getUser(id));
  if (user.isPresent()) {
    console.renderActivities(paceApi.getActivities(user.get().id));
  }
}
```

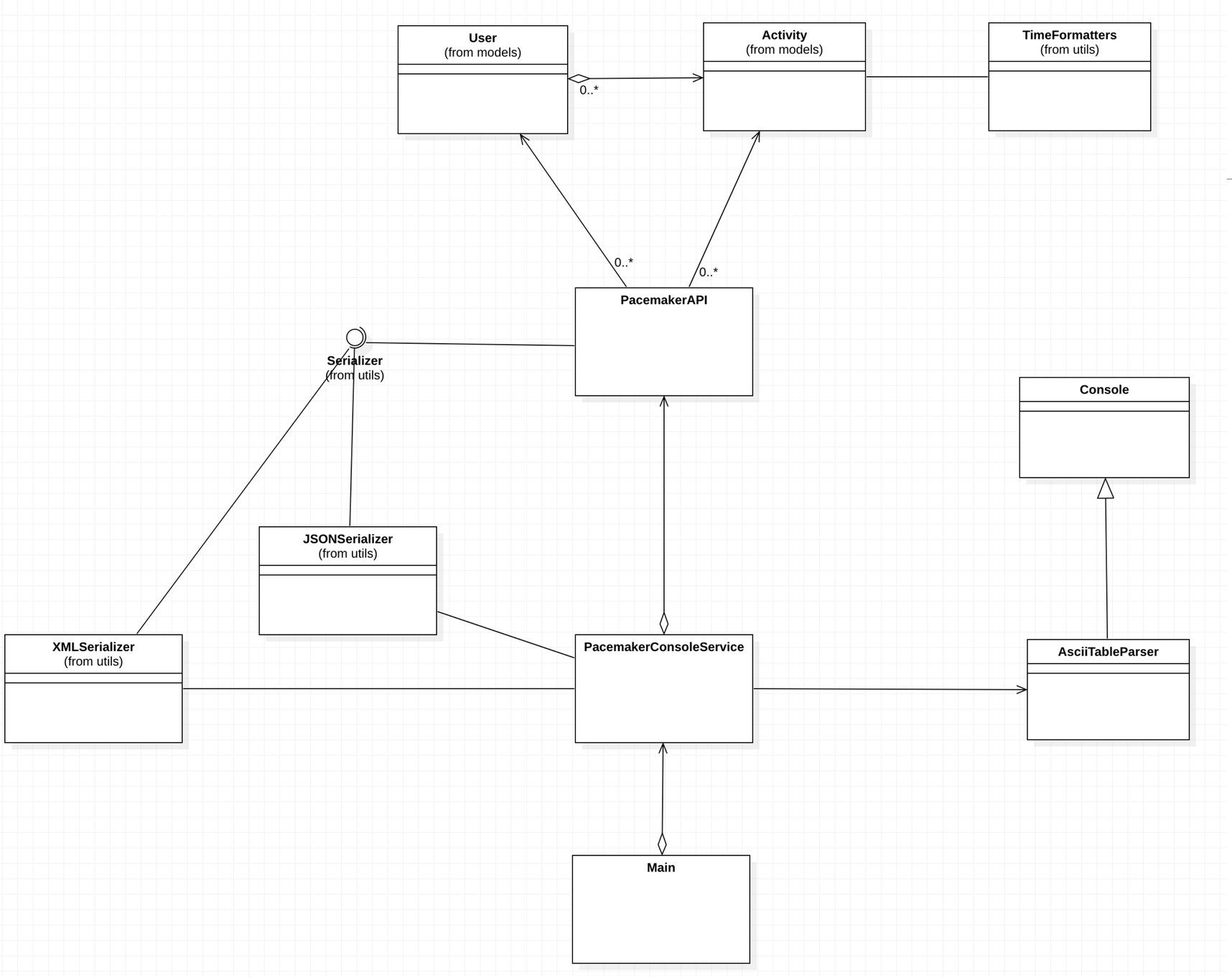
controllers -PacemakerConsoleService

```
  @Command(description="List Activities")
  public void listActivities (@Param(name="userid") String id, @Param(name="sortBy: type, location, distance, date, duration") String sortBy)
  {
    Set<String> options = new HashSet<>(Arrays.asList("type", "location", "distance", "date", "duration"));
    if (options.contains(sortBy))  {
      console.renderActivities(paceApi.listActivities(id, sortBy));
    }
    else {
      console.println ("usage : la " + options.toString());
    }
  }

  @Command(description = "Set file format")
  public void changeFileFormat(@Param(name = "file format: xml, json") String fileFormat) {
    switch (fileFormat) {
      case "xml":
        paceApi.serializer = xmlSerializer;
        break;
      case "json":
        paceApi.serializer = jsonSerializer;
        break;
    }
  }

  @Command(description = "Load activities persistent store")
  public void load() throws Exception {
    paceApi.load();
  }

  @Command(description = "Store activities persistent store")
  public void store() throws Exception {
    paceApi.store();
  }
}
```

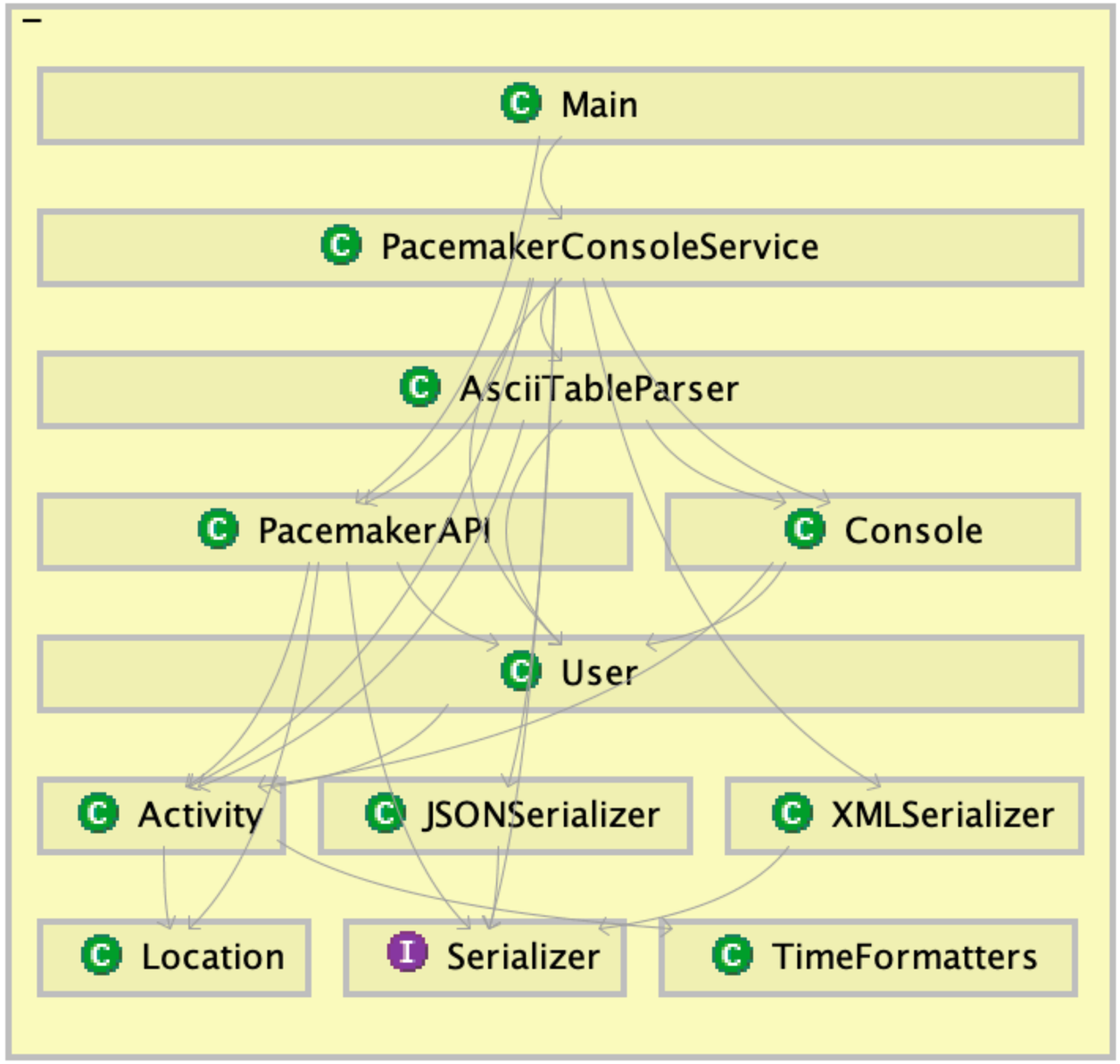
controllers - Main

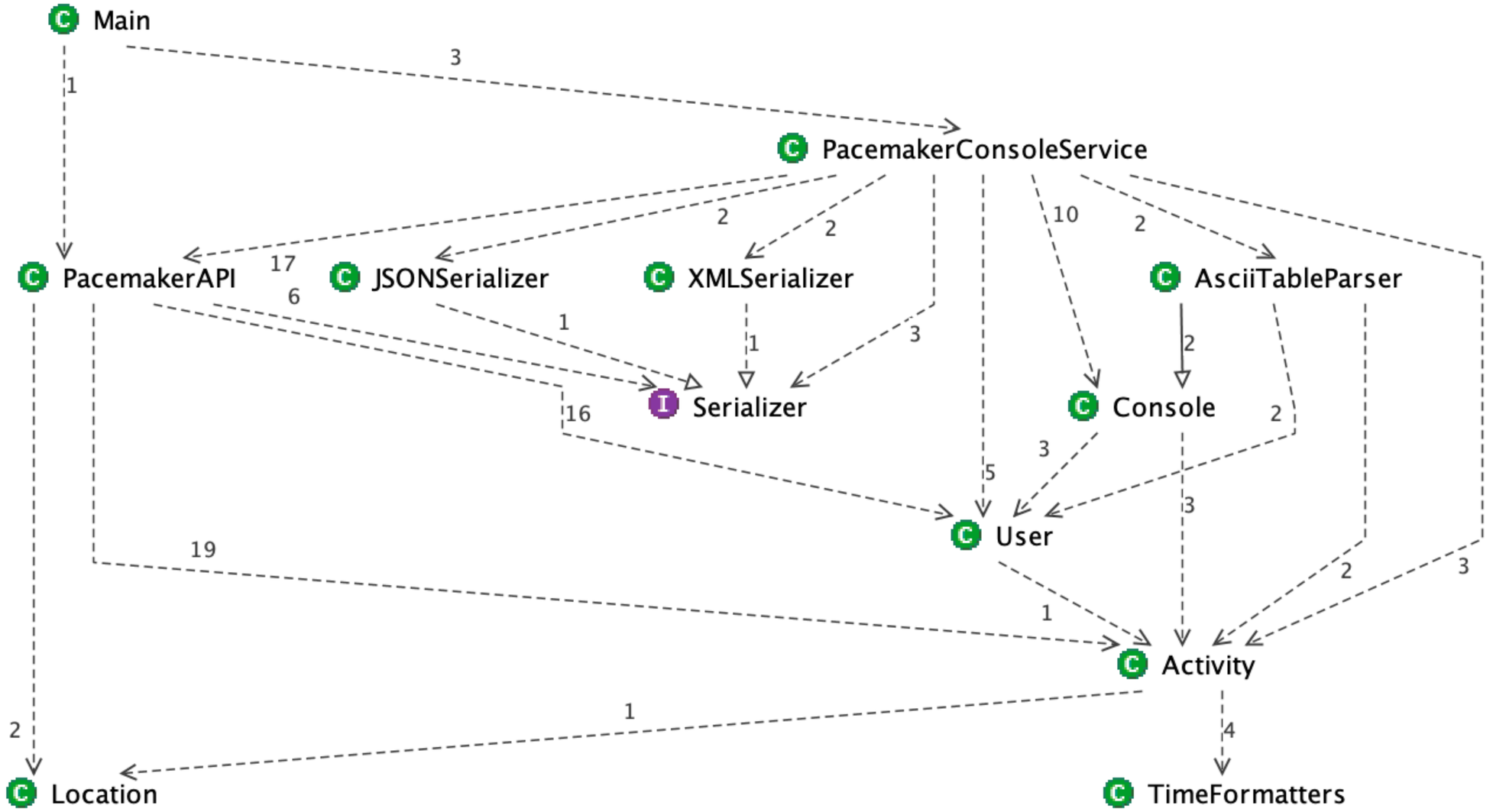
```
public class Main {
  public static void main(String[] args) throws Exception {
    PacemakerConsoleService main = new PacemakerConsoleService();

    Shell shell = ShellFactory.createConsoleShell("pm",
        "Welcome to pacemaker-console - ?help for instructions", main);
    shell.commandLoop();
    main.paceApi.store();
  }
}
```

controllers



https://structure101.com/





https://structure101.com/