Example Three: Spring

Behzad Bordbar and John Saxon

March 12, 2013

1 Task

In this worksheet, you are not expected to do any modifications to the example code. You need to read through the specification and examine the example code to see it satisfies the requirements.

In order to do this, you will need to setup your database by running the program with the parameter "create". To delete the database, you will need to run the program with the parameter "drop".

2 System Description

Acme Asset Management (AAM) maintains a set of Assets belonging to establishments such as care homes, hospitals, catering companies or residential homes for older people. This includes maintenance of electricity, plumbing, heating and various home appliances. Examples of Assets are kitchen appliances such as dishwashers and cookers, boilers, TV sets, microwave, alarms, etc. These Assets belong to the customers (care homes, hospitals,...), but AAM has a contract to look after them (Assets). There are two types of maintenance carried out by AAM, Regular maintenance, such as yearly servicing of a boiler or Reactive maintenance such as fixing of a blockage or repairing of a broken TV. All Assets which are managed by AAM are registered in the system. At this point, to keep the system small, we only focus on developing a system to allow the Reactive maintenance.

To do a maintenance job, AAM relies on tradesmen which AAM refers to as the Resources. As a result, some of the Resources are Internal (i.e. employed by the company) and some are External (i.e. contracted as needed).

3 Purpose

This document provides the specification for the provided example code. It provides an overview of the intended system and a description of the functions that are implemented in the example code. Use this document to help your understanding of the provided code.

4 Background Information

4.1 Definitions

The following definitions apply to the interpretation of this scenario:

1. Customer: the person who interacts with AAM - the employee of the company who deals with reporting of the problems, a point of contact. For example, in a care home, the customer can be one of the managers
2. Asset: An appliance or device belonging to a customer for which a maintenance agreement with AAM exists.
3. Resource: A tradesman such as a plumber or electrician who will perform the repair work on behalf of AAM.

4. Work: A maintenance task that has been initiated as the result of a Customer call and assigned to a Resource to carry out the work.
5. Work Detail: Details of the work completed by the Resource. This may relate to the complete Work or to part of the Work.

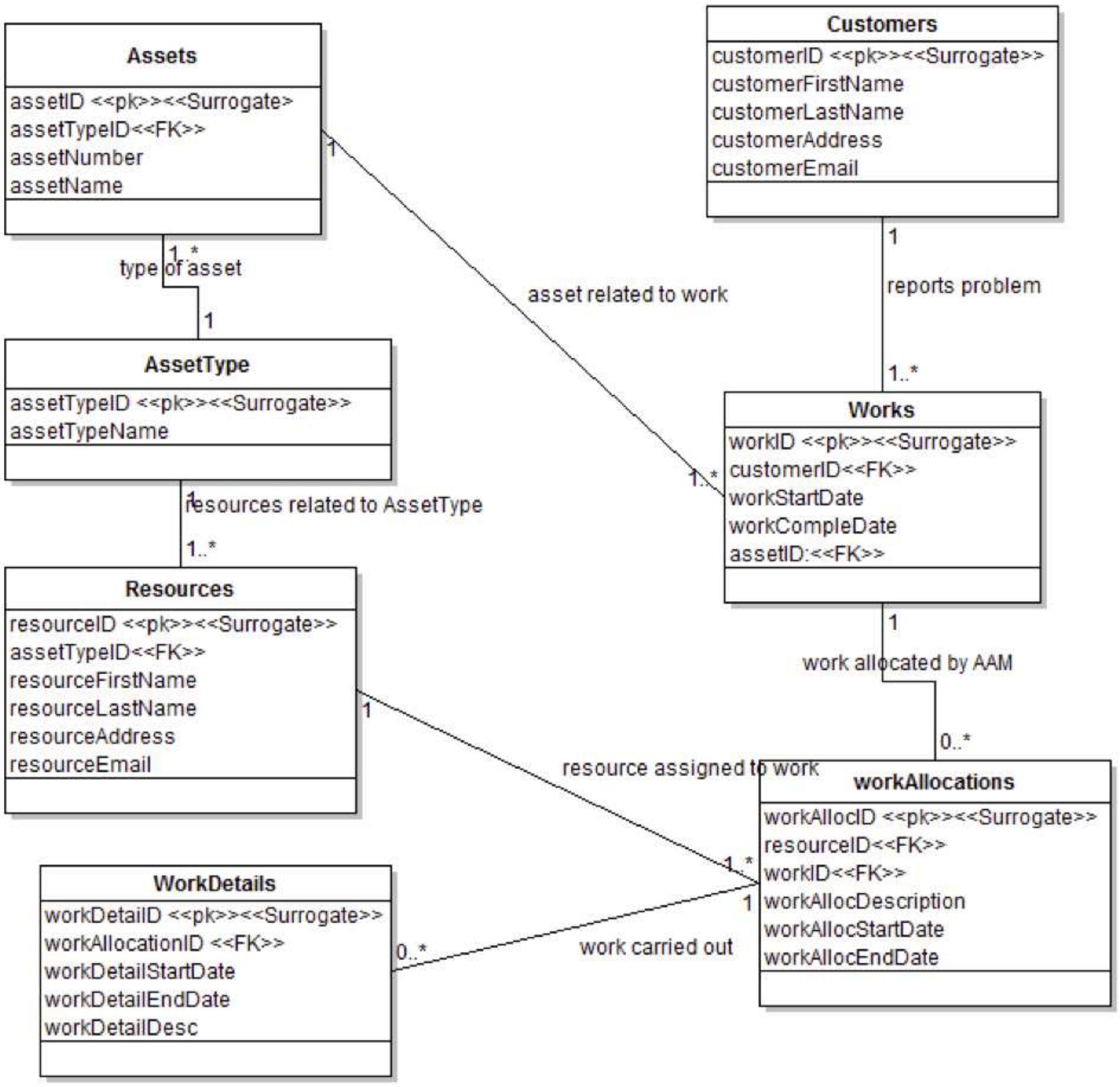
4.2 Typical Scenario

The following describes a typical scenario that is to be recorded by the system:

1. An Asset at a Customer location develops a fault (i.e. a boiler stops working).
2. The Customer telephones AAM to report the problem and to ask for maintenance to be completed.
3. AAM raises a Work, entering the date of the call as the start date for the Work.
4. The secretary at AAM, uses the system to identify the Asset that the Work relates to, and the appropriate Resources who will be responsible for ensuring the Work is completed. The Work is allocated to the Resources. A maintenance task (Work) may require one or more Resources to complete the Work.
5. The Customer can monitor the progress of the Work by logging into the system through a browser.
6. Each allocated Resource can access the system to add details of the Work (Work Detail) that needs to be completed. They may add one or more WorkDetail entries depending on the complexity of the Work.
7. When a Resource completes their allocated work, they will access the system and enter a completion date for their allocated work.
8. When all allocated work is completed, the Work will have its completion date set thus signalling the completion of the Work.

Notes The system in its current form does not keep track of the time taken and costs involved in completing a Work. As a result, it does not implement any billing related functionality. These are extensions that will be required in a later release of the system.

4.3 Database Schema



4.4 Task - Console Application

4.4.1 Implemented Example Code

The following is a list of tasks that are provided in the example Console application. You should carefully review the supplied code so that you understand what it is doing and where it is doing things.

The console application has a parameter as the first argument that represents the task that is to be completed. Each of the tasks may have additional arguments. These are dependent on the task requested.

1. Database Manipulation
   (a) Create the Database
      i. This function is called when the argument "create" is used to run the program.
      ii. It uses the Hibernate functionality to create the database tables.
      iii. It also populates the database with an initial set of data.
   (b) Drop the Database

      i. This function is called when the argument “drop” is used to run the program.
      ii. It uses the Hibernate functionality to drop the database tables.

2. Asset Type Functionality
   (a) Produces a list of the Asset Type names
      i. This function is called when the argument “ListAssetTypes” is used to run the program.
      ii. It prints a list of the Asset Type names
   (b) Add an Asset Type
      i. This function is called when the argument “AddAssetType” is used to run the program. A second argument with an Asset Type name should be provided.
      ii. If no Asset Type name is provided or the Asset Type name already exists then the program prints appropriate error messages.
      iii. If the Asset Type name is provided and not already in the database, the program adds it to the database.
   (c) Update Asset Type
      i. This function is called when the argument “UpdateAssetType” is used to run the program. Two more arguments should be provided. These are the existing Asset Type name and the new name to be assigned to the Asset Type.
      ii. If either of the additional two arguments are not provided or the existing Asset Type name is not already in the database or the new Asset Type name already exists then the program prints appropriate error messages.
      iii. If the two arguments are present and are not trapped by the conditions in step 2 then the program changes the name for the Asset Type on the database.
   (d) Delete Asset Type
      i. This function is called when the argument “DeleteAssetType” is used to run the program. A list of arguments that represent existing Asset Type names should be provided.
      ii. For each Asset Type name
         A. If no Asset Type name is provided or the Asset Type name does not exist on the database then the program prints appropriate error messages.
         B. If there are Assets or Resources associated with the Asset Type, then the program prints an appropriate error message.
         C. If there are no Assets or Resources associated with the Asset Type, then the program deletes the Asset Type from the database.

4.4.2 To be Implemented

1. Maven2
   (a) Convert this project such that it can be used with Maven2, providing handling of dependencies and a runtime environment (use JDK 1.6).
   (b) You will need the following dependencies:

Dependency Name	**Version**
spring-core	3.2.1.RELEASE
spring-orm	3.2.1.RELEASE
spring-tx	3.2.1.RELEASE
spring-aop	3.2.1.RELEASE
spring-jdbc	3.2.1.RELEASE
spring-context	3.2.1.RELEASE
postgresql	9.1-901-1.jdbc4
hibernate-core	3.6.10.Final
junit	4.11
commons-logging	1.1.1
commons-dbcp	1.4
aopalliance	1.0
aspectjweaver	1.5.4
commons-pool	1.6
javassist	3.12.1.GA

(c) **Note** you will need to remove "src" from the build path and add two source folders "src/main/*" if you are not creating a new project.

2. Asset Functionality

   (a) Produces a list of the Assets

      i. This function is called when the argument "ListAssets" is used to run the program. Two additional arguments may be provided. These are the beginning Asset number and an ending Asset Number.

      ii. If no asset numbers are provided then the program prints a list of all the Assets on the database.

      iii. If only one asset number is provided then the program prints the details for that asset number.

      iv. If two numbers are provided then the program prints all the Assets on the database that are from the first Asset number to second Asset number. If the second Asset number is less than the first Asset number then no Assets are printed.

      v. For each Asset printed, the program prints the Asset number, Asset name, Asset serial number, and the Asset Type name.

   (b) Add an Asset

      i. This function is called when the argument "AddAsset" is used to run the program. Four arguments are required. These are Asset number, Asset name, Asset serial number, and the Asset Type name.

      ii. If any of the arguments are not provided, the Asset number is already on the database or the entered Asset Type name is not on the database then the program prints appropriate error messages.

      iii. If the provided arguments pass the validation checks then the program adds the Asset to the database.

   (c) Update an Asset

      i. This function is called when the argument "UpdateAsset" is used to run the program. Five more arguments should be provided. These are the existing Asset number, a new Asset number, Asset name, Asset serial number, and the Asset Type name.

      ii. If any of the arguments are not provided, the existing Asset number is not on the database, the new Asset number is already on the database or the entered Asset Type name is not on the database then the program prints appropriate error messages.

      iii. If the provided arguments pass the validation checks then the program changes the values for the Asset on the database.

   (d) Delete an Asset

i. This function is called when the argument “DeleteAsset” is used to run the program. One or two arguments must be provided. These are the beginning Asset number and an ending Asset Number.
ii. If no Asset number is provided or the Asset numbers do not exist on the database then the program prints appropriate error messages.
iii. If only one Asset number is provided then the program deletes the corresponding Asset provided there are not Works attached to the Asset.
iv. If two Asset numbers are provided then the program attempts to delete all assets between the two Asset numbers (inclusively). All Assets must have no Associated Works attached. If any delete fails none of the deletes are committed to the database.

3. Asset Type Functionality
   (a) Revisit the list of AssetTypes and print out the Assets for each type.
4. Spring Security
   (a) Using the Spring Security examples, SpringAcegiExample and AAM, alter your application so it uses implements Spring Security.
      i. Note that the Customer class definition changes to extend another class.
      ii. Add functionality to list Work (ignore allocations).
         A. If the role of the user is *Role_admin* allow them to see all Work.
         B. If the role of the user is *Role_user* allow them to see only their work.
         C. If the role of the user is *Role_anonymous* don’t allow them to see any work.