UNIVERSITY OF JOHANNESBURG

Computer Science 2A

Practical Assignment 04

2016-03-01

Time: Deadline — 2016-03-15 12h00 Marks: 100

This practical assignment must be uploaded to eve.uj.ac.za **before** 2016-03-15 12h00. Late or incorrect submissions **will not be accepted**, and will therefore not be marked. You are **not allowed to collaborate** with any other student.

Good coding practices include a proper coding convention and a good use of JavaDoc comments. Marks will be deducted if these are not present. Every submission **must** include a batch file. See the reminder page for more details.

The Java Development Kit (JDK) has been installed on the laboratory computers along with the Eclipse Integrated Development Environment (IDE).

This practical aims to solidify your understanding of advanced Java Object Orientation

The story continues from Practical03. Aboard the SS Invictus you have managed to create a basic application to read the roster of the ship. You are able to find and contact one of the engineering crew aboard the ship. However, this specific crew member is not able to assist you in opening the doors. After going through the ships files again you come across another roster file which contains more information. This new roster file is in a similar format:

```
CREW_ID CREW_RANK CREW_SURNAME CREW_LEVEL CREW_TYPE CREW_SPECIALITY CREW_VALUE
```

CREW_ID Unique ID of the crew member, 6 characters long.

CREW_RANK Rank of the crew member, abbreviated 3 letter code.

CREW_SURNAME Surname of the crew member, can be any length.

CREW_LEVEL Experience level of the crew member, number.

CREW_TYPE Type of the crew member, any length but all capital letters.

CREW_SPECIALITY Speciality field which depends on crew type, any length.

CREW_VALUE Number dependant on the type of the crew member (explained below)

You also find information about ranks and types of the crew along with speciality associated with the crew type.

Crew Ranks:

- ADM - Admiral
- CPT - Captain
- CDR - Commander
- FLT - First Lieutenant
- SLT - Second Lieutenant
- ENS - Ensign

Crew Types:

- Combat - Trained for combat, specialised as melee specialist or ranged specialist.
- Medical - Trained for medical assistance, specialised as chemist or xenobiologist.
- Engineering - Trained for repair and maintenance, specialised as mechanist or electrician.
- Science - Trained for science experimentation, specialised as physics or chemist.
- Psychic - Trained for unknown purposes, no specialisation.

For each crew type the following additional information is available:

- Combat - `CREW_VALUE` is damage dealt amount, a number.
- Medical - `CREW_VALUE` is heal amount, a number.
- Engineering - `CREW_VALUE` is repair amount, a number.
- Science - `CREW_VALUE` is enhance item amount, a string
- Psychic - `CREW_VALUE` is psychic influence amount, a string.

Update your Java application with the following:

- Create an enumeration for the crew rank (**`E_CREW_RANK`**).
- **`CrewMember`**
  - Make the class abstract.
  - Include the enumeration for crew rank.
  - Include the level attribute.
  - Add an abstract method ***`interact`*** which takes no parameters.
  - Override the ***`toString`*** method to output a textual representation of the class attributes.
- For each type create a subclass of **`CrewMember`** (**`CrewCombat`**, **`CrewMedic`**, **`CrewEngineer`**, **`CrewScience`**, **`CrewPsychic`**)
  - Include attributes specific to the crew type.
  - Override the ***`interact`*** method to print the value amount to the console along with a descriptive message.
  - Override the ***`toString`*** method to include the extra attributes.
- Update the **`CrewRoster`** ***`readRoster`*** to include the new information in the regular expression and processing of the text file.
- Update the **`Main`** class to display the roster using the ***`toString`*** and ***`interact`*** methods of **`CrewMember`**.

Marksheet

1. Update UML class diagram [05]
2. **`CrewMember`** class
   (a) Abstract [2]
   (b) Updated crew rank [2]
   (c) Level attribute [2]
   (d) ***`interact`*** method [2]
   (e) ***`toString`*** method [7]
3. Subclasses of **`CrewMemeber`** (6 marks each). [30]
4. Update **`CrewRoster`** class - ***`readRoster`*** method [05]
5. Update **`Main`** class [05]
6. Packages [05]
7. Coding convention (structure, layout, OO design) and commenting (normal and JavaDoc commenting). [10]
8. Correct execution [25]

NB

Submissions which **do not compile** will be capped at 40%

Execution marks are awarded for a correctly functioning application and not for having some related code.

Reminder

Your submission must follow the naming convention as set out in the general learning guide.

```
SURNAME_INITIALS_STUDENTNUMBER_SUBJECTCODE_YEAR_PRACTICALNUMBER
```

Example

Surname	Berners-Lee
Initials	TJ
Student number	209912345
Module Code	CSC2A10
Current Year	2016
Practical number	P00

```
Berners-Lee_TJ_209912345_CSC2A10_2016_P00
```

Your submission must include the following folders:

- `bin` - *(Required)* Should be empty at submission but will contain runnable binaries when your submission is compiled.
- `docs` - *(Required)* Contains the batch file to compile your solution, UML diagrams, and any additional documentation files. Do not include generated JavaDoc.
- `src` - *(Required)* Contains all relevant source code. Source code must be places in relevant sub-packages!
- `data` - *(Optional)* Contains all data files needed to run your solution.
- `lib` - *(Optional)* Contains all libraries needed to compile your solution.

NB

Every submission **must** include a batch file. This batch files must contain commands which will compile your Java application source code, compile the associated application JavaDoc and run the application. **Do not** include generated JavaDoc in your submission. All of the classes/methods which were created/updated need to have JavaDoc comments.