
# Computer Science 2A

Practical Assignment 06

2016-03-22

Time: Deadline — 2016-04-05 12h00 Marks: 60

This practical assignment must be uploaded to eve.uj.ac.za **before** 2016-04-05 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 Java GUI low-level rendering

The story continues from Practical05. After building a basic GUI you are not situated to display the layout of the ship. The ship layout is stored in a text file where certain symbols have specific meaning.

Create an enumeration called **`E_TILE_TYPE`**. This enumeration will have values for each tile that the ship is made of.

Create a **`ShipLayout`** class. This class will store a 2D array of **`E_TILE_TYPE`**. Create a static ***`readShipLayout`*** method in the **`CrewRoster`** class which will return a **`ShipLayout`** instance. The ***`readShipLayout`*** method will read the layout file and fill the 2D array.

Create a **`ShipPanel`** class that extends **`JPanel`**. The **`ShipPanel`** class will override the ***`paintComponent`*** method to draw a **`ShipLayout`**. The **`ShipLayout`** must be stored as an attribute in the **`ShipPanel`** class. Each tile will be 50x50 pixels in size.

The layout of **`ShipFrame`** changes to the following:

- Left - Button panel, (Open roster, save roster and open layout)
- Right - Text area
- Center - **`ShipPanel`**

An example file is shown below:

```
12 12
_|_|____|___
=|_|=---|=--
____________
=|_|=---|=--
_|_|____|___
-|_|----|---
____________
--=---------
____________
____________
____________
____________
```

where

- **_** : Empty floor tile.
- **-** : Represents a horizontal wall tile.
- **|** : Represents a vertical wall tile.
- **=** : Represents a door tile.

The first line in the file is the size of the 2D array. Every character from the second line onwards represents a tile, the tile type corresponds to the character at that position. The **E_TILE_TYPE** enumeration stores each of these tile types.

The main package exists as **acsse.csc2a**. The sub-package structure will change as follows:

- **model** - All data classes.
- **file** - classes which handle files exclusively.
- **ui** - All GUI related classes.

The **Main** method remains unchanged.

# Mark sheet

1. **`E_TILE_TYPE`** [05]
2. **`ShipLayout`** - Attribute for 2D array of **`E_TILE_TYPE`** [05]
3. **`CrewRoster`** ***`readShipLayout`*** method [05]
4. **`ShipPanel`**
   (a) **`ShipLayout`** attribute [05]
   (b) ***`paintComponent`*** method [05]
5. Packages [05]
6. Coding convention (structure, layout, OO design) and commenting (normal and JavaDoc commenting). [10]
7. Correct execution. [20]

---

# 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.