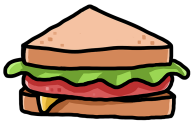

sandwico

Coding standards: Real-time Fire Escape Route

by ERP

Mathilda Bresler
u16313382@tuks.co.za

Pieter Braak
u16313382@tuks.co.za

Kateryna Reva
u17035989@tuks.co.za

Jason Louw
u16313382@tuks.co.za

Xiao Jian Li
u16099860@tuks.co.za

12 May 2019

University Of Pretoria, Hatfield
Engineering, Built environment and Information Technology

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

1 Coding standards

General coding standards pertain to how the developer writes code. Adhering to these standards improve uniform style, clarity, flexibility, reliability and efficiency of our code. The standards adhered to by our development team are listed below.

1.1 Indentation

Proper and consistent indentation is important in producing easy to read and maintainable programs. Indentation should be used to:

- Emphasize the body of a control statement such as a loop or a select statement
- Emphasize the body of a conditional statement
- Emphasize a new scope block

Consistent indentation of 4 spaces is used throughout the system.
Examples:

```
public double distanceToGoal(Node n)
    {
        double dist = 0;
        Node temp = n;
        int position = nodes.indexOf(n);
        while(position < nodes.size() - 1)
        {
            temp = nodes.get(position);
            dist += temp.distanceToNode(nodes.get(++position));
        }
        return dist;
    }
```

```
    private JSONObject sendToRTFE(JSONObject req) throws Exception {
            JSONObject Response = new JSONObject();
            try{
                switch ( (String)req.get("type")){
                    case "build":{
                        Response.put("msg",build(req));
                        break;
                    }
                    case "buildingData":{
                        Response.put("msg",BuildingToUnityString(Response));
                    }

                }
                Response.put("status", true);
            }catch(Exception e){
                if(verbose) {
                    System.out.println("CRITICAL - UNITY FAIL");
                    System.out.println(e.getMessage());
                    System.out.println(e.getStackTrace().toString());
                }
                Response.put("msg","Exception: "+e.getMessage());
                Response.put("status", false);
            }
            return Response;
        }
```

1.2 Inline comments

Inline comments explaining the functioning of the subroutine or key aspects of the algorithm shall be, and are, frequently used.
Example

```
 else { // GET or HEAD method

    if (fileRequested.endsWith("/")) {
        fileRequested += DEFAULT_FILE;
        }
    File file = new File(WEB_ROOT, fileRequested);
    int fileLength = (int) file.length();
    String content = getContentType(fileRequested);
    if (method.equals("GET")) { // GET method so we return content
    byte[] fileData = readFileData(file, fileLength);
    // send HTTP Headers
    out.println("HTTP/1.1 200 OK");
    out.println("Date: " + new Date());
    out.println("Content-type: " + content);
    out.println("Content-length: " + fileLength);
    out.println(); // blank line between headers and content, very important !
    out.flush(); // flush character output stream buffer
    dataOut.write(fileData, 0, fileLength);
    dataOut.flush();
    }
    if (verbose) {
        System.out.println("File " + fileRequested + " of type " + content + " returned");
    }
}
```

1.3 Structured Programming

Structured (or modular) programming techniques are used. GO TO statements shall not be used as they lead to "spaghetti" code, which is hard to read and maintain

1.4 Classes, Subroutines, Functions, and Methods

Methods, subroutines, and functions are kept reasonably sized. They are constrained to one function or action. If the modules become too large they shall be split into smaller subroutines.

1.5 Source Files

Source files are given meaningful names, and all linked subroutines have similar functions that are logically linked and structured.

1.6 Variable Names

Variables have mnemonic or meaningful names that convey to a casual observer, the intent of its use. Variables shall be initialized prior to its first use to eliminate chances of garbage values.

1.7 Use of Braces

In some languages, braces are used to delimit the bodies of conditional statements, control constructs, and blocks of scope. Our development team shall use either of the following bracing styles consistently: 1)

```
catch (Exception e){
//                  System.out
}
```

2)

```
catch (Exception e)
{
//                  System.out
}
```

Braces shall be used even when there is only one statement in the control block to improve readability and promote consistency.

1.8 Program Statements

Statements shall be limited to be one per line, and nested statements shall be avoided wherever possible.

1.9 Meaningful Error Messages

Error handling is very important in the system. Making error messages meaningful makes it easier to identify what errors have occurred due to which circumstances. These messages should also be displayed in ways that make it easy for review.

1.10 Design patterns

Design patters are used throughout the system to improve code maintainability and modularization. This allows for addition of features without changing the entire structure of the system.

2 File structure

- Versionnumber
  - Buildings
    - **BuildingName1**
    - **BuildingName2**
  - html
    - 404.html
    - index.html
  - src
    - ApiEndpoints
      - API
      - BuildingAPI
      - BuildingGenerationAPI
      - Database
      - WebAPI
    - Builder
      - Builder
      - BuildingManager
      - DoorBuilder
      - PersonManager
      - RoomBuilder
    - Building
      - Building
      - Door
      - Node
      - NodeType
      - Person
      - Room
      - RoomType
      - Routes
    - HTTPServer
    - Server
    - ServerTester
    - UnitTester