****

# Assessment Information/Brief 2021-22

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Module title | | | Design Patterns | | | | |
| CRN | | | 52789 | | | | |
| Level | | | 5 | | | | |
| Assessment title | | | Assignment: A JavaFX app utilising Design Patterns | | | | |
| Weighting within module | | | This assessment is worth 100% of the overall module mark. | | | | |
| Submission deadline date and time | | | 4pm Friday 29th April | | | | |
| Module Leader/Assessment set by  Dr Ian Drumm, [i.drumm@salford.ac.uk](mailto:i.drumm@salford.ac.uk), N217 | | | | | | | |
| How to submit  You are to submit your work electronically via the Blackboard website. Your submission should include…   * Your Eclipse project’s **complete directory zipped**. * A **report,** named report\_*your\_roll\_number.doc* (for example **report\_123456.doc**)which contains a title page followed by up to **6 pages** of written explanation of your software design, application of design patterns and key elements of your code. UML and other figures are recommended, though class diagrams that are auto generated by third party tools will lose marks. The title page should have exclusively your roll number (e.g. @123456), your assignment title and list of contents / subheadings. * A standalone **runnable JAR file,** named jar\_*your\_roll\_number*.jar. See the **APPENDIX on page 8** of this assignment brief for more information on how to create this. * A **video** file, named video\_*your\_roll\_number*.MP4. Using the free application <https://screencast-o-matic.com/>, or similar record your PC/Mac desktop for between 1 and 4 minutes, running your application from Eclipse. From here you can demonstrate the application’s user interface and any key aspects of code. Feel free to narrate, though this is not essential. | | | | | | | |
| Assessment task details and instructions  A local primary school wishes to use computer applications to help teach very young children to learn. You are required to create a single interactive game that teaches primary school children some aspect of science in a very colourful and fun way. Coding tasks are   * To use only Eclipse, Maven and javafx archetype as demonstrated in module. * Key objects to be drawn to the screen should be instances of subclasses of the following GameObject class.   **class GameObject {**  **protected Image img;**  **protected double x, y;**  **protected GraphicsContext gc;**    **public GameObject(GraphicsContext gc, double x, double y)**  **{**  **this.gc=gc;**  **this.x=x;**  **this.y=y;**  **}**    **public void update()**  **{**  **if(img!=null)**  **gc.drawImage(img, x, y, 30, 30);**  **}**  **}**   * You are required to implement two or more design patterns presented in the module (e.g. Delegation, Factory, Command, Singleton, MVC, Builder, etc) * Demonstrate thoughtful ‘responsibility driven design’, for example with ‘use case’ and sequence diagrams. Provide one or more class diagrams. * **Do not** use SceneBuilder or other third-party coding tools to assist in the creation of the application. Do not use third party libraries. * **Do not** present tic-tack-toe, reversi, arcade clones, text quizzes or anything else that’s obviously from a textbook, a youtube tutorial or demonstrating a lack of imagination and independent coding ability. Note any coding cynically taken from the web and presented as yours will be considered as plagiarism and the maximum penalty for unfair means will be recommended. | | | | | | | |
| Assessed intended learning outcomes  On successful completion of this assessment, you will be able to…  Knowledge and Understanding   1. demonstrate an academic understanding of OOP programming concepts 2. demonstrate a critical understanding of software design   Practical, Professional or Subject Specific Skills   1. implement a JavaFX2 application 2. apply important OOP concepts 3. leverage design patterns with suitable justification 4. discuss implementation with appropriate depth and technical language 5. communicate design diagrammatically with UML 6. undertake a critical evaluation of one’s own software design and implementation   Transferable Skills and other Attributes   1. communicate understanding in an appropriate manner 2. demonstrate scholarliness 3. take responsibility for, and ownership of, one’s own learning | | | | | | | |
| Module Aims  The aim of this module is to develop knowledge and understanding of the  application of generic pattern based strategies for resolving common high-level  problems in object orientated design. Students will develop an appreciation of  problem recognition and categorization and learn how established pattern based  solutions can be identified and applied as solutions to these problems. | | | | | | | |
| Word count/ duration  Your report should be between 3 and 6 pages, there is no word count but do not use a font size below 10 pts. Your name and roll number must be on the report. Note do not waste space with near empty title pages or similar large areas of white space. A title page and contents page is also recommend and is not included with the 6 page limit. | | | | | | | |
| Feedback arrangements  You can expect to receive feedback via blackboard / grade centre with marks for respective components and associated comments. | | | | | | | |
| Support arrangements  You can obtain extra support for this assessment via staff surgeries as specified on the timetable.  askUS  The University offers a range of support services for students through [askUS](http://www.askus.salford.ac.uk/).  Good Academic Conduct and Academic Misconduct  Students are expected to learn and demonstrate skills associated with good academic conduct (academic integrity). Good academic conduct includes the use of clear and correct referencing of source materials. Here is a link to where you can find out more about the skills which students require <http://www.salford.ac.uk/skills-for-learning>.  **Academic Misconduct is an action which may give you an unfair advantage in your academic work. This includes plagiarism, asking someone else to write your assessment for you or taking notes into an exam. The University takes all forms of academic misconduct seriously. You can find out how to avoid academic misconduct here** [**https://www.salford.ac.uk/skills-for-learning**](https://www.salford.ac.uk/skills-for-learning)**.**  Assessment Information  If you have any questions about assessment rules, you can find out more [here](http://www.salford.ac.uk/qeo/AssessmentPolicy).  Personal Mitigating Circumstances  If personal mitigating circumstances may have affected your ability to complete this assessment, you can find more information about personal mitigating circumstances procedure [here](https://sss.salford.ac.uk/).  Personal Tutor/Student Progression Administrator  If you have any concerns about your studies, contact your Personal Tutor or your Student Progression Administrator. | | | | | | | |
| Assessment Criteria | | | | | | | |
|  | **70-100%** | | **60-70%** | **50-60%** | **40-50%** | **0-40%** |
| **User Experience**  **Worth 10/100** | Outstanding user experience which fits the brief perfectly.  Makes exceptional use of the user interface, graphics and sound.  Is simple, intuitive, fun and pretty. | | An enjoyable and appropriate user experience fitting the assignment brief.  Makes good use of the user interface, graphics and sound.  Is simple, intuitive, fun, pretty. | Good, competent user experience.  Is simple, intuitive, fun, pretty. | Competent though limited user experience.  Working but lacking in either simplicity, ease of use, fun or attractive presentation. | Very poor user experience.  Lacking in simplicity, ease of use, fun or attractive presentation.  Runnable jar doesn’t run on another university PC. |
|  | **70-100%** | | **60-70%** | **50-60%** | **40-50%** | **0-40%** |
| **Design (evident in report and coding)**  **Worth 30/100** | Exceptionally well thought out and understood application of design patterns  Outstanding functionality.  Exceptionally elegant.  Impeccable choice of classes to encapsulate their respective data and functionality  Impeccable use of interfaces.  Exceptional design.  Evidence student has taken responsibility for own learning with application of relevant knowledge acquired beyond notes. | | Appropriate and justifiable application of design patterns.  Thoughtful and elegant design as reflected in coding.  Appropriate choice of classes to encapsulate data and functionality  Appropriate use of interfaces.  Evidence student has taken responsibility for own learning with application of relevant knowledge acquired beyond notes. | Competent application with good functionality fitting the assignment brief.  Good, evidenced design.  Some good application of design patterns. | More thought with regards design could have been shown.  Good though limited functionality.  Some application of design patterns. | No real thought of design.  No working application or with limited functionality.  Project directory incomplete or un-importable.  Not fitting the brief. |
| **Coding**  **Worth 20/ 100** | Impeccable presentation of coding with appropriate comments, naming conventions, indenting, etc.  Clearly building on notes and exercises presented throughout the module. Whilst making appropriate choices of knowledge to apply.  Exceptional creative programming with impressive functionality.  A good understanding and creative use of the JavaFX2 API beyond that presented in notes. | | Thoughtful and tidy presentation of coding.  Clearly building on notes and exercises presented throughout the module. Whilst making appropriate choices of knowledge to apply.  Good creative programming with impressive functionality.  A good understanding and creative use of the JavaFX2 API. | Thoughtful and tidy presentation of coding (formatting, choice of class, method and field names, etc).  Sensible (not excessive) use of comments to explain OO programming employed.  Making good use of examples in notes. | Could improve presentation of coding.  Using comments but could be more appropriate or consistent. | Untidy presentation of coding.  Coding difficult to read and understand.  Code doesn’t build or work well enough to meet the assignment briefing.  Application doesn’t load and work straight away when imported into assessor’s equivalent IDE.  Runnable jar file doesn’t work because student has incorrectly referenced or packaged resources in code, or used incompatible or overly large resources. Student has not submitted a short video instead. |
| **Report**  **Worth 40/100** | An exceptionally well-presented report.  Exceptionally good understating of topics presented in notes and exercises evident.  Exceptionally good critique and evaluation of own code.  Exceptional use of diagrams to communicate key information.  Report is very easy to read and understand for typical programmer but with an excellent use of technical and academic language  Evidence student has read beyond notes to support learning and showing an academic researcher’s level of scholarliness.  An impeccable sense of the academic as well as practical expectations of a degree level module. | | A well-presented report showing evidence of the thoughtful and sensible use of object-oriented techniques and responsibility driven design.  Very good critique and evaluation of own code.  All diagrams (e.g. Class Diagram) and images are meaningful, readable and relevant. All diagrams stick to conventions and communicate key information.  Very good understating shown of topics presented in notes and exercises evident.  Evidence student has read beyond notes to support learning.  Report is easy to read and understand but with appropriate use of technical and academic language. | Fair presentation and some presentation of design and evaluation.  Reasonably good understating of topics presented in notes and exercises evident.  Reasonable critique and evaluation of own code and design.  No auto generated diagrams. | Report too brief or too wordy with irrelevant waffle.  Report could be better presented.  No meaningful critique or evaluation.  A poor use of diagrams that are either too crowded, unreasonably untidy, inappropriate, not sticking to conventions or too sparse to be meaningful. | Brief, irrelevant or unreadable report.  Has auto generated diagrams.  Has screen grabs of code rather than copied in well explained and selected snippets of particularly key or important coding. |

|  |
| --- |
|  |
| The following is a generic guide to percentage marks with respect to level of performance.  90-100 Outstanding  80-89 Excellent  70-79 Very Good  60-69 Good  50-59 Fair  40-49 Adequate  30-39 Unsatisfactory  20-29 Poor  10-19 Very Poor  0-9 Extremely Poor  Reassessment  If you fail your assessment, and are eligible for reassessment, you will need to resubmit on or before August. For students with accepted personal mitigating circumstances for absence/non submission, this will be your replacement assessment attempt.  The task for reassessment will be the same as the original task. |

**APPENDIX**

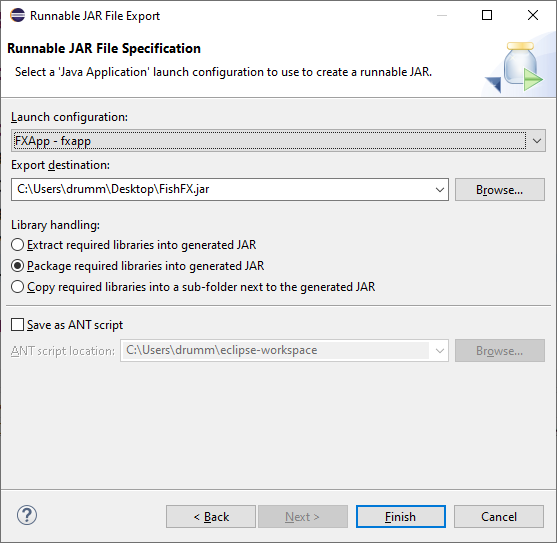
Ensure you have already installed the Java Development Kit and Eclipse as instructed in the week one worksheet and video for the Design Patterns module.

## Creating a runnable Jar file

Because we are using a Maven archetype, bundling the application as a runnable jar for standalone execution is relatively easy. Right click on, for example, the **uos.*your\_fx\_app*** package in the Package Explorer and choose

**Export->Runnable Jar**

And click [Next].



* From the resulting dialog box, click on the **Launch Configuration** combo box to choose the name of your app.
* Browser to a suitable **Export Destination** such as your desktop.
* Ensure **Package required libraries into JAR** is checked.
* Click [Finish]

Hence you have a standalone application on your desktop to run. Double click on it to try it out, or if this does not work run from a terminal or dos command shell with **java -jar *YourFXApp*.jar**

**Note JAR files do not run on computer suite PCs due to a legacy java run time on the path of these systems, required for other modules. However, your JAR will run on your PC or Mac if you had installed the JDK and Eclipse as instructed at the start of this module. Hence the jar will run on the assessors PCs.**