**CSE-271: Object-Oriented Programming**

**Exercise #8**

Max Points: 20

|  |  |
| --- | --- |
| **Name:** | Zach Clouse |

|  |  |
| --- | --- |
| Pin | For your own convenient reference – You should first save/rename this document using the naming convention **MUid\_Exercise8.docx** (example: amjadm\_Exercise8.docx) prior to proceeding with this exercise. |

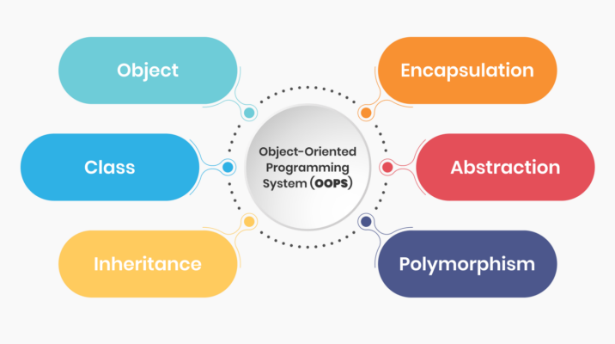
|  |
| --- |
| **Objectives**: The objectives of this exercise are to:   1. Review the concepts of Graphical User Interfaces (GUI) 2. Review basics of Swing class hierarchy (JFrame, JPanel, JButton)    1. Develop a custom JComponent via inheritance & polymorphism 3. Gain some familiarity layout managers (GridLayout) 4. Gain familiarity with GUI builder – Eclipse’s Window Builder tool   Fill in answers to all of the questions. For some of the questions you can simply copy-paste appropriate text from Eclipse output into this document. You may discuss the questions or seek help from your neighbor, TA, and/or your instructor. |

# Part #0: One time setup of Eclipse (IDE) – Only if needed

|  |  |
| --- | --- |
| Eclipse Logo A2 by dj-fahr on DeviantArt | We already configured Eclipse’s source formatter and Checkstyle plug-in as part of Lab #1. If your Eclipse is not configured (because you are using a different computer) then use the instructions from Lab #1 to configure Eclipse. |

# Part #1: GUI basics and Object-oriented programming

*Estimate time: < 30 minutes*



**Background**: Object-oriented Programming (OOP) is a programming paradigm that is heavily used for developing GUI libraries such as Swing and gaming libraries such as Unity. OOP enables developing an flexible and unambiguous (*i.e.*, intention is clearly communicated between programmers who develop libraries versus those who use the libraries).

It is also important to remember that building a GUI essentially involves same basic concepts:

* Creating objects and calling methods with suitable arguments
* Using existing methods for basic problem-solving.

**Exercise**: Briefly (2-to-3 sentences each) respond to the following questions regarding generic concepts of Graphical User Interface (GUI)

1. What is a Graphical User Interface (GUI)?

|  |
| --- |
| A GUI is an alternative for text based I/O, using graphics of the modern computer. It can use different input devices such as a mouse, keyboard, etc. |

1. Briefly (2-to-3 sentences) describe 1 advantage offered by a GUI

|  |
| --- |
| GUI’s are user friendly. It is much easier for a user to interact with a GUI. |

1. Briefly (2-to-3 sentences) describe 1 disadvantage of GUIs

|  |
| --- |
| GUI’s are less efficient. It requires more steps to carry out the same exact task. |

1. Referring to the following Wikipedia page <https://en.wikipedia.org/wiki/Xerox_Alto>, briefly (2-to-3 sentences) describe the importance of the Xerox Alto.

|  |
| --- |
| The Xerox Alto was the very first computer with a graphical operating system. This led way to the future of computing. |

1. GUIs use pixels in many different colors to create a picture. The colors of pixels are determined based on a combination of 8-bits (or 256 distinct values) of red, green, and blue (RGB) colors (sadly, RAObot’s eyes are only 3-bit color). The colors can be represented as RGB triplets or correspondingly as hex codes. Complete the following table using information from <https://htmlcolorcodes.com/color-names> (the first one is already completed to illustrate an example):

|  |  |  |
| --- | --- | --- |
| **Name** | **RGB** | **HTML** |
| Miami Pantone Red | rgb(152, 0, 46) | #98002e |
| Light Salmon | **rgb(255, 160, 122)** | **#FFA07A** |
| Deep Pink | **rgb(255, 20, 147)** | **#FF1493** |
| Lemon Chiffon | **rgb(255, 250, 205)** | **#FFFACD** |
| Medium Aquamarine | **rgb(102, 205, 170)** | **#66CDAA** |
| Ghost White | **rgb(248, 248, 255)** | **#F8F8FF** |

1. What is the event-driven programming (or approach) that is used in GUIs?

|  |
| --- |
| Event-driven programming is when all actions are represented as events occurring in the system. So, events are stored in a queue and processed one after another. |

1. What does mean to “*fire an event*”? When are events fired?

|  |
| --- |
| Firing an event is when an event is generated. This happens when the user interacts with a GUI. |

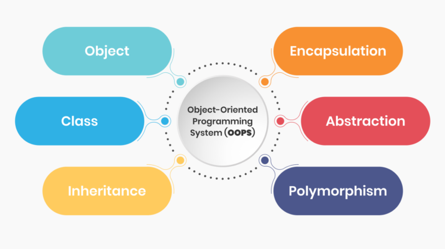
1. Briefly (2-to-3 sentences) describe 1 advantage offered by a GUI-builder program

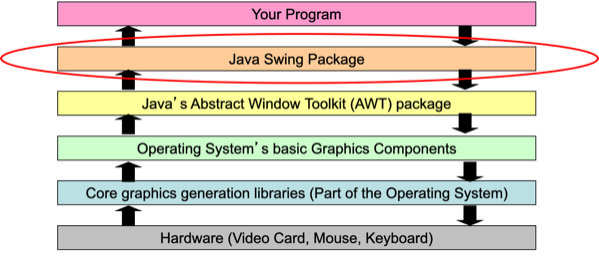
|  |
| --- |
| GUI builders let the programmer drag and drop components. This can enable rapid prototyping and have a low learning curve. |

1. Briefly (2-to-3 sentences) describe 1 disadvantage of GUI-builder programs

|  |
| --- |
| GUI builders cannot handle complex GUIs and interactions. Also, they can cause for the code to be clunky. |

# Part #2: Java-Swing and GUI programming

*Estimate time: < 30 minutes*

**Background**: Object-oriented Programming (OOP) is a programming paradigm that is heavily used for developing GUI libraries such as Swing and gaming libraries such as Unity. OOP enables developing a flexible and unambiguous (*i.e.*, intention is clearly communicated between programmers who develop libraries versus those who use the libraries).

It is also important to remember that building a GUI essentially involves same basic concepts:

* Creating objects and calling methods with suitable arguments
* Using existing methods for basic problem-solving.

**Exercise**: Briefly (2-to-3 sentences each) respond to the following questions involving the Java-Swing GUI library

1. What is a layout manager?

|  |
| --- |
| A layout manager decides the size and placement of each component in a container. The common types are border, flow, and grid layouts. |

1. What is a “functional interface”?

|  |
| --- |
| A functional interface is one that has exactly one method. This makes their parameters and returns values easy to see. |

1. Rewrite the following methods that involve anonymous classes to use the more concise lambda syntax:

|  |  |
| --- | --- |
| **Anonymous class syntax** | **Corresponding lambda syntax** |
| doIt.addActionListener(**new**  ActionListener() {  @Override    public void  actionPerformed(ActionEvent e) {  System.out.println(“Click!”);  }  }); | doIt.addActionListener((e) ->  { System.out.println(“Click!”); }); |

# Part #3: Working with Eclipse’s Window Builder

*Estimated time: < 30 minutes*

**Background**: GUI-builders are software tools that are used to ease development and/or rapid-prototyping of GUI. Window Builder is a Java-Swing GUI-builder that is available for Eclipse. It provides a visual approach for laying out components and adding actions to components. As the GUI is developed, the corresponding source code for it is automatically generated/updated by Window Builder.

|  |  |
| --- | --- |
| **Exercise**: In this part of the exercise, you are expected to install and use Window Builder to develop the GUI shown in the adjacent screenshot. |  |

|  |  |
| --- | --- |
| Clapper board | There is a video on Canvas demonstrating the use of Eclipse’s Window Builder. It may be useful to quickly review these videos as part of this exercise. |

# Part #4: Submit to Canvas via CODE plug-in

*Estimated time: < 5 minutes*

**Exercise:** You will be submitting the following files on Canvas as normal submissions:

1. This MS-Word document saved as a PDF file – **Only submit PDF file**.
2. The Java source file for the GUI that you developed using Window Builder

Ensure you actually complete the submission on Canvas by verifying your submission (after you submit)