**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **10/06/2020** | **Name:** | **Lavanya B** |
| **Course:** | **Kicad** | **USN:** | **4al17ec043** |
| **Topic:** | **A hands on kicad with simple project- schematic design** | **Semester & Section:** | **6th A** |
| **Github Repository:** | **Lavanya-B** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report**  **A hands on kicad with simple project- schematic design**  **Mounting holes**  **Mounting holes are one very PCB design, butthe reisvery little documentation**  **about this subject matter. Another issue that interferes with standardization is Imperial Unit ASNI hardware and ISO Metric hardware.So we’re going to have to explain both unit systems for clarity. Start with the basic fundamentals that both unit systems have in common.**  **Create library**  **Libraries are created from within the footprint editor. To create a new library you have two options. Either use the New Library entry of the File menu or of the right. The library name should be unique within the chosen library table. It should also communicate what footpirnts to expect within it to your target audience. Do not include special chars inside your library name as it might create problems in some platforms.**  **After creating the library you will need to add the library to the global or project library table. Adding it to the global table will make this lib visible to all your projects. The project library table only adds it to the current project.**  **Create a PCB footprint**  **A footprint describes the interface between the circuit board and the component it self. This is often called the land-pattern. At the least it needs to contain so called pads. It is also suggested to at least include the part outline and part identifiers on the silk and fab layers. If a footprint is specialized for a single component then include the part number and the manufacturer name. For generic footprints include all identifying parameters.**  **Steps to create footprint**   * **Create pads** * **Defining the components height and area** * **Add silk screen** * **Saving the design**   **Create the pads- Pads are the representation of exposed copper features to which the leads of the component will be soldered. Exposed copper means that in addition to defining the copper area the pad also needs to define a hole in the so called solder mask. Typically this cut-out is larger than the copper feature. This is achieved using the clearance settings. The hole components there is also the drill defined.**  **In addition the pad number must also be set to allow the connection between footprint and symbol.**    **Defining components height and area- In this step, we need to define the height and area occupied by the component. We also need to define the component type. To access this information, select your new component footprint from the Footprints list, and click the Edit button. From here, you’ll be able to enter these three pieces of information. By default, the component type will be Standard; this is the value we would want for this**  **component.Othercomponents,suchasmechanicalelementsandno-BOM**  **components, will not be standard components and should be assigned the**  **appropriate component type.**  **Add silk screen- The silkscreen is what will be printed onto the board. The colour of it will most likely be white. This highly depends on your preferences, the space available and on your manufacturers restrictions. The outline is normally drawn outside the nominal dimensions of the part. It should also not overlap any exposed copper features.**  **Saving the design- The final step is to create your component is to name and save it so you can add it to your component, which also includes the schematic symbol. You will want to make the name unique and searchable so you can easily locate it.** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **10/06/2020** | **Name:** | **Lavanya B** | |
| **Course:** | **Java** | **USN:** | **4al17ec043** | |
| **Topic:** | **Programming core JAVA** | **Semester & Section:** | **6th A** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Report**  **Programming core JAVA**  **Multidimensional array**  **Multidimensional arrays are array that contain other arrays. The two-dimensional array is the most basic multidimensional array.**  **To create multidimensional arrays, place each array within its own set of square brackets.**  **Eg:**  **public class Program {**  **public static void main(String[] args) {**  **int[ ][ ] myArr = { {1, 2, 3}, {4}, {5, 6, 7} };**  **myArr[0][2] = 42;**  **int x = myArr[1][0];**    **System.out.println(x);**  **}**  **}**  **Classes and objects**  **A class describes what the object will be, but is separate from the object itself.**  **In other words, classes can be described as blueprints, descriptions, or definitions for an object. You can use the same class as a blueprint for creating multiple objects. The first step is to define the class, which then becomes a blueprint for object creation.**  **Each class has a name, and each is used to define attributes and behavior.**  **Methods**  **Methods define behavior. A method is a collection of statements that are grouped together to perform an operation. System.out.println() is an example of a method.**  **Eg:**  **class MyClass {**  **static void sayHello() {**  **System.out.println("Hello World!");**  **}**  **public static void main(String[ ] args) {**  **sayHello();**  **}**  **}**  **Return type**  **The return keyword can be used in methods to return a value.**  **we could define a method named sum that returns the sum of its two parameters.**  **we defined the return type before we defined the method name. For our sum method, it is int, as it takes two parameters of the type int and returns their sum, which is also an int.**  **Eg:**  **class MyClass {**  **static int sum(int val1, int val2) {**  **return val1 + val2;**  **}**  **public static void main(String[ ] args) {**  **int x = sum(2, 5);**  **System.out.println(x);**  **}**  **}**  **Getter and setter**  **Getters and Setters are used to effectively protect your data, particularly when creating classes. For each variable, the get method returns its value, while the set method sets the value.**  **Getters start with get, followed by the variable name, with the first letter of the variable name capitalized.**  **Setters start with set, followed by the variable name, with the first letter of the variable name capitalized.**  **Eg:**  **public class Vehicle {**  **private String color;**  **public String getColor() {**  **return color;**  **}**  **public void setColor(String c) {**  **this.color = c;**  **}**  **}**  **Constructor**  **Constructors are special methods invoked when an object is created and are used to initialize them.**  **A constructor can be used to provide initial values for object attributes.**  **- A constructor name must be same as its class name.**  **- A constructor must have no explicit return type.**  **Eg:**  **public class Vehicle {**  **private String color;**    **Vehicle() {**  **this.setColor("Red");**  **}**  **Vehicle(String c) {**  **this.setColor(c);**  **}**    **public void setColor(String c) {**  **this.color = c;**  **}**    **public String getColor() {**  **return color;**  **}**  **}**  **public class Program {**  **public static void main(String[] args) {**  **Vehicle v1 = new Vehicle();**    **Vehicle v2 = new Vehicle("Green");**    **System.out.println(v2.getColor());**  **}**  **}** | | | |