

# PROJECT 2:

## Problem 1: Simple ASCII Art

### Overview

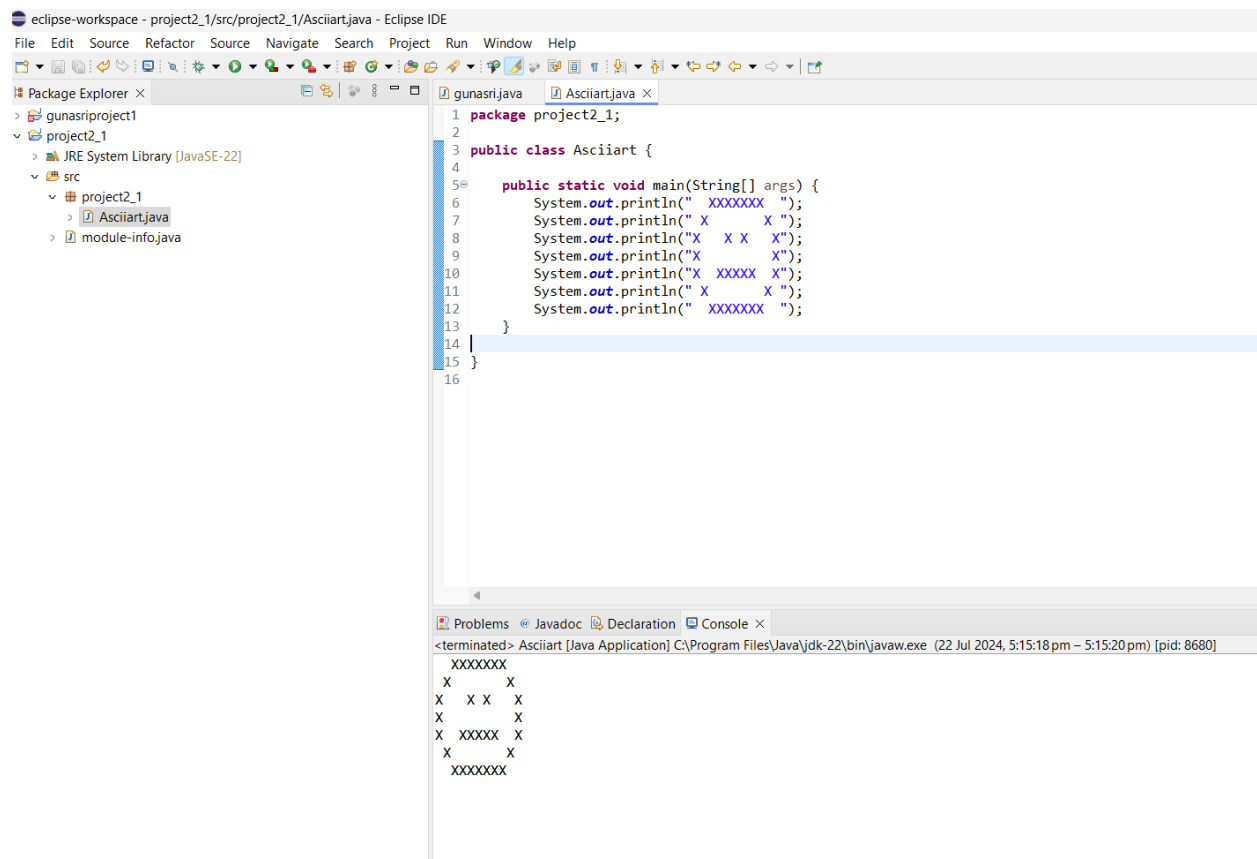
Using text to create a picture is known as ASCII art. In section 2, we made an ASCII art cat. In this practice, you'll use print statements to recreate the image above.

### Task

Use 8 print statements to recreate the smiley face above. Your art will rely on only a single character, besides space, such as X or #.

The ProblemSet2\_1 project is available to help you get started.

## PROJECT 2.1



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure: gunasriproject1, project2\_1, JRE System Library [JavaSE-22], src, project2\_1, Asciiart.java, and module-info.java. The main editor window shows the code for Asciiart.java:

```
1 package project2_1;
2
3 public class Asciiart {
4
5     public static void main(String[] args) {
6         System.out.println(" XXXXXX ");
7         System.out.println(" X  X X  X ");
8         System.out.println("X  X X  X");
9         System.out.println("X  XXXX  X");
10        System.out.println("X  XXXX  X");
11        System.out.println(" X  X  X ");
12        System.out.println(" XXXXXX ");
13    }
14 }
15 }
16 }
```

The Console window at the bottom shows the output of the program:

```
<terminated> Asciiart [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (22 Jul 2024, 5:15:18 pm - 5:15:20 pm) [pid: 8680]
XXXXXX
X  X  X
X  X X  X
X  XXXX  X
X  XXXX  X
 X  X  X 
XXXXXX
```

## PROJECT 2.2:

### Task

Use print statements to create your own beautiful original ASCII art. Use comments to describe what your image is depicting.

It's ok for your art to rely on only a single character, besides space, such as X or #. But you're encouraged to use a few different characters in your design, like in the cat example from class.

Your art must also:

- Use at least 8 print statements
- Be at least 8 characters wide
- Use at least 20 characters that aren't space

You're welcome to create another cat. However, this image must look significantly different from the example used in class. Similarly, you're welcome to create another face, but it must look significantly different from the face in the previous practice (it's way too easy to turn the smiley face into a frowny face).

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project named 'gunasriproject1' with a source folder 'src' containing files 'gunasriproject1', 'gunasri.java', 'project2\_2.java', 'Program Files', and 'module-info.java'. The main editor window shows the code for 'project2\_2.java', which defines a package 'gunasriproject1' and a class 'project2\_2' with a 'main' method. The 'main' method prints an ASCII art representation of a cat's head. The console at the bottom shows the output of the program, which is the ASCII art cat image.

```

1 package gunasriproject1;
2
3 public class project2_2 {
4     public static void main(String[] args) {
5         // cat image
6         System.out.println("  /\_/\  ");
7         System.out.println(" //  \\// ");
8         System.out.println(" //  \\// ");
9         System.out.println(" //  /\_/\  ");
10        System.out.println("((  ))");
11        System.out.println("====  V  ====");
12        System.out.println("====((  ||  ))====");
13        System.out.println(" (      ) ");
14        System.out.println("  (____)  ");
15    }
16 }
17

```

```

<terminated> project2_2 [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (22 Jul 2024, 7:06:14 pm - 7
  /\_/\
 //  \\//
 //  \\//
 //  /\_/\
((  ))
====  V  ====
====((  ||  ))====
 (      )
  (____)

```

## PROJECT 2.3

### DOCUMENT TO CREATE SNAKE BOX FACTORY SOFTWARE DESIGN

#### 1. Object: SnakeBox

##### Properties:

Dimensions: The size of the box, typically including length, width, and height.

MaterialQuality: The quality of the cardboard used to make the box.

SnakeType: The specific type of snake that will be placed inside the box.

##### Behaviors:

CalculateBoxVolume(): Computes the volume of the box based on its dimensions.

CheckMaterialQuality(): Assesses the quality of the cardboard to ensure it meets standards.

AssignSnakeType(SnakeType type): Assigns a specific snake type to the box and adjusts any related settings.

## **2. Object: Snake**

### **Properties:**

Species: The species of the snake.

Size: The size of the snake, which may influence the size of the box required.

HealthStatus: The current health status of the snake.

### **Behaviors:**

ChangeHealthStatus(String status): Updates the health status of the snake.

Grow(Size newSize): Adjusts the size attribute of the snake as it grows.

GenerateReport(): Creates a report on the snake's current health and characteristics.

## **3. Object: Order**

### **Properties:**

OrderID: A unique identifier for each order.

CustomerDetails: Information about the customer placing the order, such as name and address.

OrderStatus: The current status of the order (e.g., Processing, Shipped, Delivered).

### **Behaviors:**

UpdateOrderStatus(String newStatus): Changes the status of the order to reflect its current state.

GenerateInvoice(): Creates an invoice based on the details of the order.

TrackShipment(): Provides tracking information for the shipment of the order.