Instructions:

1. Click on File -> Make a copy

Include your IPO Chart with your code in Eclipse IDE: We will discuss and demonstrate this in the computer lab.

IPO CHART

Program name:	**Nim Game (Nim)**	
INPUT	**PROCESS**	**OUTPUT**
Hint: What will the user input?	Hint: What is the program going to do with the input information?	Hint: What will the screen display after user input?
The user will input how many stones they want to take (*int*) **Not inputs but Constants/Limits:** ● Minimum starting stones (15) ● Maximum starting stones (30) ● Maximum stones to take (3)	1. Initialization: ● Generate a random starting number of stones for the pile. (*int*, *Random*) ● Initialize the game state variables, including *stones* (*static int*) and the *userLost* flag (*boolean*). 2. User Turn Handling (*handleUserTurn*): ● Read the user's input using the *Scanner* object. ● Validate the move (check if it's an integer, between 1 and 3, and legal) using *isValidEntry* (*boolean*). ● Update the *stones* count by subtracting the user's choice. ● Determine if the user's move caused *stones == 0* (user loses). 3. Computer Turn Handling (*handleComputerTurn*): ● Generate a random, legal move (*computerChoice*,	1. Game State/Prompts: ● The current number of remaining stones. (*String*) ● Prompts asking the user for their move. (*String*) 2. Moves & Status: ● Messages reporting the number of stones the computer took. (*String*) ● Messages for invalid user input or illegal moves. (*String*) 3. Result/Win Condition: ● Final message declaring the winner: "The player beats the computer!" or "The computer beats the player!". (*String*)

	int) using *drawStones*. ● Update the *stones* count by subtracting the computer's choice. 4. Game Flow (Main Loop): ● Alternate turns until the winning condition (*stones == 0*) is met. ● Identify the winner (the player who did not take the last stone).	