**IPO**

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| <Form load>;  List Value; | Randomize the value of the cases by randomly select values from list Value. | <Nothing> |
| Select one case (when handleSelection() is called);  List Cases; | Locks case by storing the value in the case to a new global variable “Selected” and then set the value of the case in the list Cases to 0. | Global Variable Selected; |
| <Button press> (Selecting other cases);  List Cases;  List BankTime; | If the status is selection, then call handleSelection();  If the status is open, then call handleOpen();  If the status is finish, do nothing;  If the status is final, call handleFinal(); | Output depends on each functions. |
| Function bank() is called;  List BankTime;  Input from message box | Determine the value of bank offer by calling function calcOffer(), and then show up a message box with yes and no.  If yes is pressed, then the game finishes and the user gets that money.  If no is pressed, then the game continues. | Output varies:  YES – The value that the user can get  NO – Nothing, redirection to updateStatus() |
| Function calcOffer() is called;  List Cases; | Loop through everything in the Cases, determine the number of non-negative values and the sum of all non-negative values, then find the bank offer by the average (sum / number) multiplied by 0.85. | Float bankOffer; |
| Function updateState() is called;  List Cases;  Variable status; | Check the status and sets the new status accordingly.  Before all those, call updateLabel().  If status is load, then instruct user to select one to keep and update state to selection;  If status is selection, then instruct user to select ones to open and update state to open;  If status is open, then loop through the list Cases and determine the number of unopened, unselected case. If that number is in list BankTime, then switch to bank(). Otherwise, do nothing;  If status is bankfinish, then update status to open;  If status is finish, display a message box that tells the user the game is already finished;  If status is final, update status to finish. | Updated status; |
| Choose one to reveal value (Function handleOpen() is called) | Check the value in the case.  If it is a negative value, show up a message box that indicates that case is already open.  If it is 0, show up a message box that the participant is already eliminated, and end the game immediately.  If it is a positive value, set its value to negative (by multiplying it by -1), then update the label for newest value.  After all those, call function updateState() | Revealed value |
| Last round, choose one to open and keep (Function handleFinal() is called);  List Cases; | Check if the value is valid.  Open the case, sets its value to negative.  Call updateState() | Revealed value |
| updateLabel() is called;  List Cases; | Updates the label according to the list Cases | Updated label value. |