SHEN UI UML:

GUI 0..1

```
public:

    j1GUI();
    ~j1GUI();

    bool Awake(pugi::xml_node& config);
    bool Start();
    bool PreUpdate();
    bool Update(float dt);
    bool PostUpdate();
    bool CleanUp();

    bool Save(pugi::xml_node&) const;
    bool Load(pugi::xml_node&);

    void Update_Position(j1GUIelement* element, iPoint position, iPoint localPosition);
    j1GUIelement* ADD_ELEMENT(GUItype type, j1GUIelement* parent, iPoint Map_Position, iPoint Inside_Position, bool interactable, bool enabled, SDL_Rect section, char* text = nullptr, j1Module* listener = nullptr,
bool X_drag = false, bool Y_drag = false, SCROLL_TYPE scrollType = SCROLL_TYPE::SCROLL_NONE, bool decor = false);
    p2List<j1GUIelement*>   GUI_ELEMENTS;
```

0..N UI ELEMENT:

```
public:

    j1GUIelement() { };
    ~j1GUIelement();

    virtual bool Awake(pugi::xml_node& config) { return true; };
    virtual bool Start();
    virtual bool PreUpdate() { return true; };
    virtual bool Update(float dt) { return true; };
    virtual bool PostUpdate() { return true; };

    virtual bool CleanUp() { return true; };
    virtual bool Save(pugi::xml_node& file) const { return true; };
    virtual bool Load(pugi::xml_node& file) { return true; };

    bool OnAbove();
    virtual void OnClick();
    virtual void OnRelease();
    virtual void Dragging();
    void Draw();

public:

    j1GUIelement* parent = nullptr;
    j1Module* listener = nullptr;
    GUItype type = GUItype::GUI_ERROR;
    iPoint Map_Position = { 0,0 };
    iPoint Inside_Position = { 0,0 };

    SDL_Color color = { 255,255,255 };
    char* text = nullptr;
    bool enabled = false;
    bool interactable = false;

    bool X_drag = false;
    bool Y_drag = false;

    bool above = false;
    bool focus = false;

    bool decorative = false;

    SDL_Rect rect = { 0,0,0,0 };
    SDL_Texture* texture = nullptr;
```

0..N BUTTON:

```
public:

    j1GUIButton();
    ~j1GUIButton();

    bool Awake(pugi::xml_node&);
    bool Start();
    bool PreUpdate();
    bool Update(float dt);
    bool PostUpdate();
    bool CleanUp();

    bool Load(pugi::xml_node&) { return true; };
    bool Save(pugi::xml_node&) const  const { return true; };

    void Dragging();
    void MovingIt(float dt);
    void OnRelease();

private:

    j1GUIelement* label = nullptr;
    iPoint Drag = { 0,0 };

    bool dragging;
```

0..N IMAGE:

```
public:

    j1GUIimage();
    ~j1GUIimage();

    bool Awake(pugi::xml_node&);

    bool PreUpdate();
    bool PostUpdate();

    bool CleanUp();

    bool Load(pugi::xml_node&) { return true; };
    bool Save(pugi::xml_node&) const  const { return true; };
```

0..N LABEL:

```
public:

    j1GUIlabel();
    ~j1GUIlabel();

    bool Awake(pugi::xml_node&);
    bool Start();

    bool PreUpdate();
    bool Update(float dt);
    bool PostUpdate();
    bool CleanUp();

    bool Load(pugi::xml_node&) { return true; };
    bool Save(pugi::xml_node&) const  const { return true;
};
```

0..N SCROLL BAR:

```
public:

    j1GUIscrollBar(SCROLL_TYPE type);
    ~j1GUIscrollBar();

    bool Awake(pugi::xml_node&);
    bool Start();

    bool PreUpdate();
    bool Update(float dt);
    bool PostUpdate();

    bool CleanUp();

    bool Load(pugi::xml_node&) { return true; };
    bool Save(pugi::xml_node&) const  const { return
true; };
    void ScrollLimits();


    j1GUIelement* Button = nullptr;
    SCROLL_TYPE Type =
SCROLL_TYPE::SCROLL_NONE;

    float Value;
```

0..N INPUT BOX:

```
public:

    j1GUIinputBox(char* text);
    ~j1GUIinputBox();

    bool Awake(pugi::xml_node&);
    bool Start();

    bool PreUpdate();
    bool Update(float dt);
    bool PostUpdate();

    bool CleanUp();

    bool Load(pugi::xml_node&) { return true; };
    bool Save(pugi::xml_node&) const  const { return true; };

    void OnClick();
    void OnRelease();
private:

    j1GUIelement* string = nullptr;
    j1GUIelement* background = nullptr;
```