Implementation 2

Cohort 1, Team5:
Lewis Mcshane
Adam Howard
Morgan Davis
Road Gibbons
Zijun Zou
Muhidin Muhidin

The new architecture reflects the code base in how the three separate UML diagrams of the concrete architecture correspond with the different packages of the project directory. Since there is the main folder, then the two packages: HUD and OperativeAI. This means that the code more accurately depicts the architecture in a clear manner since they have a similar layout.

Moreover, we've implemented the missing features of infiltrator abilities and a pause menu as portrayed in the updated architecture. This displays that the code base and architecture have stayed consistent as they have been updated.

Then, the new code implements the requirements since, as seen below, the requirements that were left unfulfilled after last term have been coded in. This is omitting the demo, which we have explained the circumstances for below.

1) Infiltrator Abilities: FR_OPER_ABILITY & UR_OPERATIVE_ABILITIES

We have had to implement the infiltrator abilities as they were not implemented when we picked up the project. This was the bulk of what we needed for the full-functionality of the game.

a) Invisibility

This ability causes the infiltrator to have a chance of becoming invisible when they attack a system. This code is found within the draw method in the Operative class (Implemented code begins after line 150).

```
142     @Override
143     public void draw(Batch batch, float parentAlpha) {
144       // Prevents problems caused by race conditions (where they are killed and then draw is run)
145       if (dead){
146         return;
147       }
148       if (!Constants.paused){
149         update(batch);
150       }
151
152       // Has the chance of becoming invisible when attacking a system
153       // Happens hackingHide number of times (currently 5) with a chance of randomHide (currently 1/3)
154
155       // Initialises the Random class
156       Random rand = new Random();
157       // Assigns a random integer with bound 2 to randomHide
158       randomHide = rand.nextInt( bound: 2);
159       // Checks whether the Infiltrator is hacking and whether they have hid the maximum number of times
160       // Also checks if the Infiltrator should hide with randomHide
161       if((isHacking) && (hackingHide > 0) && (randomHide == 1)){
162         // Reduces number of hides left by 1 each time it is called
163         hackingHide = hackingHide - 1;
164         // stableHide is used to ensure the Infiltrator doesn't stay hidden once finished hacking
165         stableHide = true;
166       }
167       else if(!stableHide || !isHacking){
168         // Draws the Infiltrator only if stableHide is false or the Infiltrator is not hacking
169         batch.draw(image,  x: getX() - hitboxOffset,  y: getY() - hitboxOffset, image.getWidth(), image.getHeight());
170       }
171     }
```

b) Avoidance

This ability causes the infiltrator to have the chance to gain spain and flee from the player after being hit. This code is found within the update method in the Operative class (New implemented code begins after line 328).

```
326         if (nodeNum >= currentPath.getCount()){
327
328           // Has a chance of increasing speed and assigning new target
329           // This chance occurs once the Infiltrator engages in combat
330           // It can happen runAway times (currently 3) at a probability of runAwayRandom (currently 1/5)
331
332           // Assigns a random integer with bound 4 to runAwayRandom
333           int runAwayRandom = rand.nextInt( bound: 4)+1;
334           // Checks whether the Infiltrator has fled the maximum number of times
335           // Runs if runAwayRandom is equal to 1
336           if(runAway>0  && runAwayRandom == 1){
337             // Chooses a new system to attack
338             chooseTarget();
339             // Resets the currentPath to the new target system for the Infiltrator to attack
340             currentPath = pathfinder.findPath(map.gridPos(getX()),map.gridPos(getY()), target.gridX,target.gridY);
341             // Increases the movement speed of the Infiltrator to 4
342             moveSpeed = 4f;
343             // Reduces the number of times the ability can be used by 1
344             runAway = runAway - 1;
345           }else{
346             // Resets the movement speed to 1.2 if the ability is not used
347             moveSpeed = 1.2f;
348             // Resets the currentPath to the player for combat
349             currentPath = pathfinder.findPath(map.gridPos(getX()),map.gridPos(getY()), map.gridPos(player.getX()),map.gridPos(player.getY()));
350           }
351           nodeNum = 0;
352         }
353         move();
354       }
355     }
```

c) Communications Jam

This ability gives a chance to hide the player's HUD after attacking an infiltrator for a given period of time. The new code is that which includes the variables jamTime and jammed.

```
153     @Override
154     public void draw(){
155         if (!jammed) {
156             super.draw();
157         } else{
158             jamTime -= Gdx.graphics.getDeltaTime();
159             if (jamTime <= 0){
160                 jammed = false;
161             }
162         }
163         //Set the current values
164         systemsHealthBar.setCurrentValue(Systems.systemsRemaining.size());
165         operativesHealthBar.setCurrentValue(Operative.remainingOpers);
166
167         this.act();
168
169         //If the player moves off a teleporter pad hide the dialogue
170         if (!teleporterDialog.isPlayerTouchingTeleporter()){
171             teleporterDialog.hide();
172         }
173     }
```

```
47      /**
48       * Amount of time a communication jam occurs for
49       * Whether the HUD is jammed
50       */
51      private float jamTime = 1;
52      private boolean jammed = false;
```

```
194     public void Jam(float time){
195         jamTime = time;
196         jammed = true;
197     }
```

2) Pause Menu UR_UX

We decided to add a pause menu, since it would make the game more accessible and user friendly. This was implemented as a part of the HUD, where you can press escape and a menu will pop up.

```java
public PauseMenu(AuberGame game){
    this.game = game;

    setup();
}

public void setup() {

    //Create the stage and allow it to process inputs. Using an Extend Viewport for scalablity of the product
    stage = new Stage(new ExtendViewport(Gdx.graphics.getWidth(),Gdx.graphics.getHeight()));

    //Create the table and expand it to fill the window
    Table table = new Table();
    table.setFillParent(true);

    //Create the logo and add it to the table
    Texture logoTexture = new Texture(Gdx.files.internal( path: "img/menu/auberLogo.png"));
    Image logo = new Image(logoTexture);
    table.add(logo).pad(10).fillY().align(Align.center);
    table.row();

    //Create the start game button, add it to the table with its click event
    ImageButton.ImageButtonStyle playStyle = new ImageButton.ImageButtonStyle();
    playStyle.up = new TextureRegionDrawable(new TextureRegion(new Texture(Gdx.files.internal( path: "img/menu/playButtonInactive.png"))));
    playStyle.down = new TextureRegionDrawable(new TextureRegion(new Texture(Gdx.files.internal( path: "img/menu/playButtonActive.png"))));
    playStyle.over = new TextureRegionDrawable(new TextureRegion(new Texture(Gdx.files.internal( path: "img/menu/playButtonActive.png"))));
    ImageButton playButton = new ImageButton(playStyle);
    table.add(playButton).center().pad(5);
    table.row();
```

```java
    playButton.addListener(new InputListener() {
        @Override
        public boolean touchDown(InputEvent event, float x, float y, int pointer, int button) {
            //As per libGDX docs this is needed to return true for the touchup event to trigger
            return true;
        }

        @Override
        public void touchUp(InputEvent event, float x, float y, int pointer, int button) {
            menuSelect.play( volume: 0.2f);
            game.setScreen(new GameScreen(game));
            Constants.paused = false;
        }
    });

    //Create the menu button, add it to the table with its click event
    ImageButton.ImageButtonStyle menuStyle = new ImageButton.ImageButtonStyle();
    menuStyle.up = new TextureRegionDrawable(new TextureRegion(new Texture(Gdx.files.internal( path: "img/menu/menuButtonInactive.png"))));
    menuStyle.down = new TextureRegionDrawable(new TextureRegion(new Texture(Gdx.files.internal( path: "img/menu/menuButtonActive.png"))));
    menuStyle.over = new TextureRegionDrawable(new TextureRegion(new Texture(Gdx.files.internal( path: "img/menu/menuButtonActive.png"))));
    ImageButton quitButton = new ImageButton(menuStyle);
    table.add(quitButton).center().pad(5);
    table.row();

    quitButton.addListener(new InputListener() {
        @Override
        public boolean touchDown(InputEvent event, float x, float y, int pointer, int button) {
            //As per libGDX docs this is needed to return true for the touchup event to trigger
            return true;
        }

        @Override
        public void touchUp(InputEvent event, float x, float y, int pointer, int button) {
            menuSelect.play( volume: 0.2f);
            game.setScreen(new TitleScreen(game,  isMusicPlaying: false));
            Constants.paused = false;
        }
    });

    stage.addActor(table);
}

public void on() { Gdx.input.setInputProcessor(stage); }

public void draw(float delta){
    stage.act(delta);
    stage.draw();
}
```

3) Demo FR_DEMO

We were not able to implement a working demo within the game itself, however we have included a video demo on the website. We couldn't complete the demo in-game as LibGDX doesn't seem to have video compatibility for playing video.