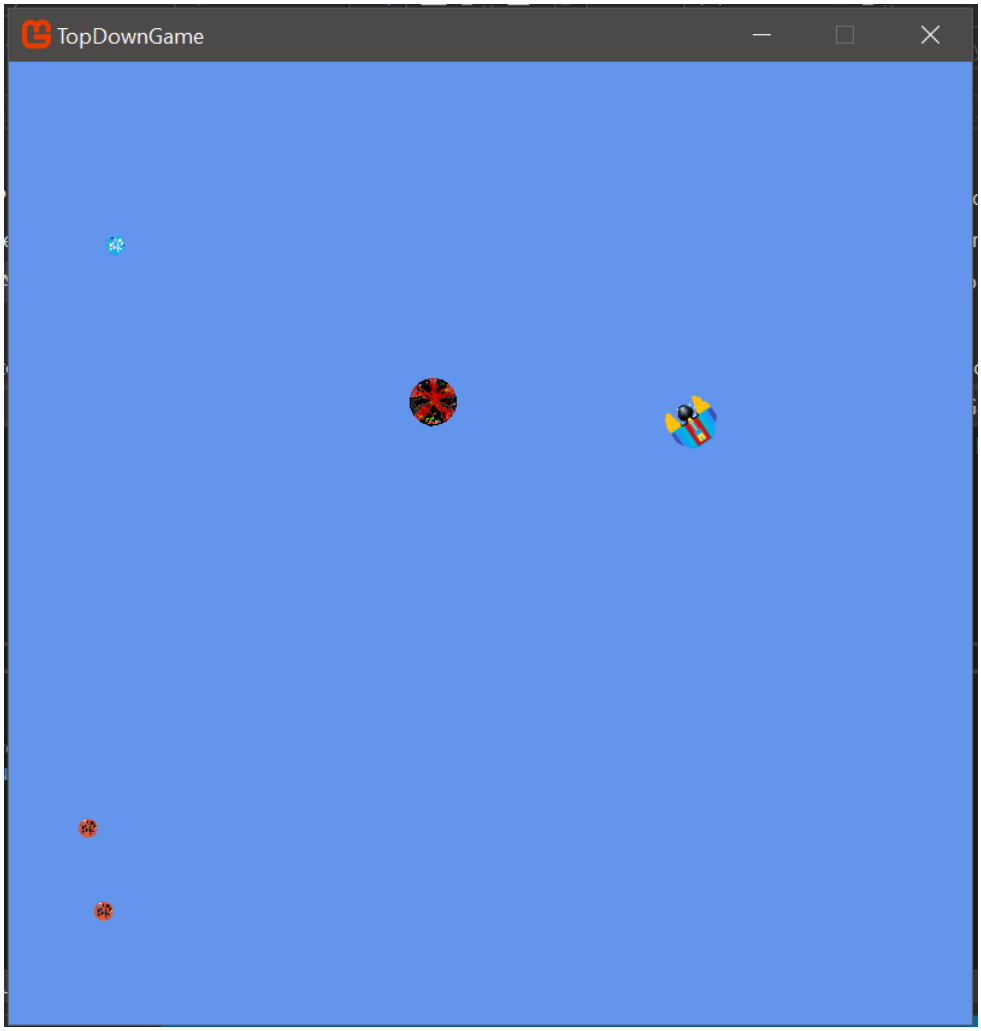
Design Overview for <<TopDownGame>>

Name: SM Ragib Rezwan
Student ID: 103172423

Summary of Program

Basically it's a TopDownShooter/ survival type game. Player will move around the screen, listening to music and avoiding different types of enemies (each with different behaviour) that spawn while trying to shoot them down using different types of projectiles. There is a timer counting down and once timer reaches 0, if player is still alive, then player wins and gets a certain number of points and happy music plays. Otherwise, player dies getting nothing and sad music plays.
(currently left to add timer, points and mainmenu, victory and defeat screens)



(it will look something like this. Will add background and touch it up later)

Required Roles

Roles that I plan on using:

1. Entity
2. Player
3. ActiveGameLogic
4. VictoryScreen
5. DefeatScreen
6. Program
7. Game1
8. SoundPlayer
9. Song1
10. SoundEffect1
11. IAudio
12. Enemy
13. EnemyNormal
14. EnemyHoming
15. EnemySpawner
16. Projectile
17. ProjectileManager
18. ProjectileNormal
19. ProjectileSlow
20. CollisionDetector
21. EnemyProjectileCollisionResponder
22. EnemyPlayerCollisionResponder
23. KeyboardController
24. ICommand
25. DownCommand
26. LeftCommand
27. NulCommand
28. QuitCommand
29. RightCommand
30. UpCommand
31. Global (to store all constant stuff)

Table 1: <<Entity>> details

Responsibility	Type Details	Notes
Entity	Constructor, Takes in parameters `Vector2 position, Vector2 dimension, string address, Game game`	It is used as a parent class that other objects can use as template to make themselves. Sets position, dimension and also how to load texture from address using game.Content.load.texture2D>
Pos	Property, returns Vector2_pos and sets Vector2 _pos	It is used to get and set the position of the objects created
Dim	Property, returns	It is used to get and set the dimension of the object

	Vector2 _dim and sets Vector2 _dim	
Speed	Property, returns float _speed and sets float _speed	It is used to get and set the speed of the object
Health	Property, returns int _health and sets int _health	It is used to get and set the health of the object
IsExpired	Property, returns int _IsExpired and sets int _IsExpired	It is used to get and set the IsExpired value of the object
SoundPlayer	Property, returns SoundPlayer _soundPlayer and sets SoundPlayer _soundPlayer	It is used to get and set the sounds of the object (if they need any)
Rotation	Property, return float _rotation and sets float _rotation value	It is use to get and set the rotation of the object
Update	Virtual method, Takes in parameter GameTime gametime, Returns void	Mainly will let the child classes override this to update themselves in their desired way
Draw	Virtual method, Returns void	This mainly sets the conditions to draw the object properly with all the conditions set in a single line (see the comment written to understand what each part does). When other object will call base.draw(), they will be able to utilize this method to draw themselves

Table 2: <<Player> details (may need to add a method for point storage later on)

Responsibility	Type Details	Notes
Player	Constructor, Takes in parameters `Vector2 position, Vector2 dimension, string address, Game game`	It is used entity to make itself. Also it is going to set player speed, health, collideddamage, lastshot, direction and also projectile manager
LastShot	Property, returns double _lastshot and sets double _lastshot	It is used to get and set the lastshot value
CollideDamage	Property, returns int _collideddamage and sets int _collideddamage	It is used to get and set the collided damage value (ie the damage player will do to enemy when collided with it)
ProjectileManager	Property, return ProjectileManager _projectileManager and sets ProjectileManager _projectileManager	It is use to get and set the Projectile Manager
HitEnemy	Method, returns void	Reduce player _health by 1, if health

		below zero, then play player killed sound by soundplayer and load defeat screen (screen hasn't been set yet)
GoRight	Method, returns void	Changes player position to right by using its speed
GoLeft	Method, returns void	Changes player position to left by using its speed
GoUp	Method, returns void	Changes player position to up by using its speed
GoDown	Method, returns void	Changes player position to down by using its speed
Update	override method, Takes in parameter GameTime gametime, Returns void	Mainly will tell _projjectilemanager to update itself using gametime, and also will use base to update itself as well
Draw	override method, Returns void	Mainly will tell _projjectilemanager to draw itself, and also will use base to draw itself as well
GetInstance	Static method, Takes in Parameter Game game, returns Player	It will create an instance of the player class using singleton design pattern

Table 3: <<ActiveGameLogic> details

Responsibility	Type Details	Notes
ActiveGameLogic	Constructor, Takes in parameters `Game game`	It is going to get a player instance, create enemyspawner, store game and _game, create soundplayer,create keyboardinput, create collisiondetector objects
Update	Virtual method Takes in GameTime gameTime, Returns void	As long as player Isexpired is false, It is used to do all of these: • tell enemyspawner to update itself, get position of cursor, use it to find player position difference and thus find direction player is rotating and facing, (will move playerdirection and position to a different method in same class and call that method here later on) • Logic to determine when to let player shoot and what (ie mouse click, shootdelay and type of projectile shot), (for this part, maybe will move It to different class for mouse input) • Tell collision detector to detect whether enemy collided with player or projectile or not For all cases: tell player to update itself, tell keyboardinput to

		update itself
Player	<<readonly>>Property, returns Player	It is used to get player
Draw	virtual method, Returns void	Tells player to draw itself, tells enemyspawner to draw itself

Table 4: <<VictoryScreen> details (still didn't make this one, basically it will load an image saying player won, play a music in background, with points player earned being stated in text (may need to pass in game to get the text available for use, also may modify player to pass point))

Responsibility	Type Details	Notes
VictoryScreen	Constructor,	
Draw	method,	

Table 5: <<DeathScreen> details (still didn't make this one, basically it will load an image saying player diead, play a music in background)

Responsibility	Type Details	Notes
DeathScreen	Constructor,	
Draw	method,	

Table 6: <<Program> details

Responsibility	Type Details	Notes
Main	Constructor,	Gets a game instance and tells game to run

Table 7: <<Gane1> details (will add logic for loading difference screens later on)

Responsibility	Type Details	Notes
Game1	Constructor,	It is used to make a new instance of graphics manager, set height and width of game, set the root directory for image and audio to content folder, make mouse visibility true and create instance of soundplayer
Initialize	Override method, Return void	It is used to initialize game1 using parent class game (prebuilt in library)
LoadContent	Override method, Return void	Loads in new instance of spritebatch, makes new instance of activegamelogic,plays background song for active music (will add conditions for loading other screen and background music here later on)
Update	Override method, Takes in parameter GameTime	It makes activelogic update itself using gametime parameter and also it updates itself using the same parameter with help of parent class game (will add conditions for loading other screen and background

	gameTime Returns void	music here later on)
Draw	Override method, Takes in parameter GameTime gameTime Returns void	Clears device screen and loads a colour, (will alter this part to add a background image later on), Draws all sprites stored in global spritebatch, tells activelogic to draw itself,ends spritebatch drawing, draws itself using gametime parameter
GetInstance	Static method, returns Game1	It will create an instance of the game1 class using singleton design pattern

Table 8: <<SoundPlayer> details

Responsibility	Type Details	Notes
SoundPlayer	Constructor, Takes in parameter Game game	Creates dictionaries for song1 and soundeffect1, creates all needed song1 and effect1 using helper class and stores them in dictionaries for central management and easy use.
PlaySong	Method, Takes parameter string Returns void	It is used to search though songlibrary dictionary to play a certain song using helper class
PlaySoundEffect	Method, Takes parameter string Returns void	It is used to search though soundEffectlibrary dictionary to play a certain soundEffect using helper class
StopMusic	Static method, Returns void	Stops whatever background song is being played

Table 9: <<Song1>> details

Responsibility	Type Details	Notes
Song1	Constructor, Takes in parameter string address, Game game	Creates a song instance by using game.Content.Load <song> and passing in where song is being stored
Play	Method, Returns void	It is used to lower volume of mediaplayer, using it to play a song, make it repeat over when finished

Table 10: <<SoundEffect1>> details

Responsibility	Type Details	Notes
SoundEffect1	Constructor, Takes in parameter string address, Game game	Creates a soundEffect instance by using game.Content.Load <soundeffect> and passing in where soundeffect is being stored

Play	Method, Returns void	It is used to play the sound effect

Table 11: <<IAudio> interface details

Responsibility	Type Details	Notes
Play	Interface method that returns void	Mainly used it to ensure song1 and soundeffect always have Play method that returns void. (This can be used to put them both in a single dictionary instead of putting in two different dictionary, but I am not doing that as I prefer to keep song and soundeffects separate)

Table 12: <<Enemy> details (may need to add a field and property for point later on) (currently kepts oldmovetime,x,y here as it can be useful when adding other enemy behaviours later on (if need be))

Responsibility	Type Details	Notes
Enemy	Constructor, Takes in parameters `Vector2 position, Vector2 dimension, string address, Game game`	It is used entity to make itself. Also it is going to set _isExpired as false, _oldmovetime as 0, and create an instance of _soundplayer
OldMoveTime	Property, returns double _oldMoveTime and sets double _oldMoveTime	It is used to get and set the oldMoveTime
X	Property, returns float x and sets float x	It is used to get and set the x value (will be used to change enemy position's x value later on using logic in child classes)
Y	Property, returns float x and sets float y	It is used to get and set the y value (will be used to change enemy position's y value later on using logic in child classes)
IsExpired	Property, returns bool _isExpired	It is used to know whether enemy has expired or not
WasHit	Method, takes in parameter int returns void	Decreases health with an int value that is passed to the method. If health is less or equal to 0, turn _isExpired into true and tell soundplayer to play enemykilled sound
Update	override method, Takes in parameter GameTime gametime, Returns void	Will update itself using its parent class if _isExpired is set to false.
Draw	override method, Returns void	Will draw itself using its parent class if _isExpired is set to false.

Table 13: <<EnemyNormal> details (may need to add point later on)

Responsibility	Type Details	Notes

EnemyNormal	Constructor, Takes in parameters `Vector2 position, Vector2 dimension, string address, Game game`	It will follow template in parent class to make itself. Also it is going to set its own speed and health values
CalculateNewMove	Method Returns void	It is used to create a random x and y value between -5 to 5, to give enemy randomness in movement
Update	override method, Takes in parameter GameTime gametime, Returns void	Will update itself using its parent class. Then it will check if certain time has been passed since it last made correction to its movement (using a move timelimit set in global). If so, it will CalculateNewMove (changing its x and y parts of vector added to its position) and store the currentgametime into oldmovetime to recalculate later on. Then it adds vector of x and y part multiplied with its speed to position to change its position. Also, if it hits screen (enemyHitScreen) then alter x and y velocity to make it hit wall and return back (or if its unlucky enough, it can be get stuck vibrating on the wall for a while if it gets low random values, making it easy for player to kill it)
EnemyHitScreen	Method, Returns Bool	Condition used to see whether it hits wall or not from any side

Table 14: <<EnemyHoming> details (may need to add point later on)

Responsibility	**Type Details**	**Notes**
EnemyHoming	Constructor, Takes in parameters `Vector2 position, Vector2 dimension, string address, Game game`	It will follow template in parent class to make itself. Also it is going to set its own speed and health values And also make an instance of activelogic
Update	override method, Takes in parameter GameTime gametime, Returns void	Will update itself using its parent class. Then it update its position by using position of player in activelogic and its own speed (beneficial for player as now if enemy is far, it wil run to player but if its close it will slowly creep towards the player making it easier to player to shoot it)

Table 15: <<EnemySpawner> details

Responsibility	Type Details	Notes
EnemySpawner	Constructor, Takes in parameters `Game game`	It makes a new enemy list, sets oldspawntime to 0 create a list for removeEnemy, pass game to its own _game field
EnemyList	Property, returns List<Enemy>_enemyList and sets List <Enemy> _enemylist	It is used to get and set Enemylist
SpawnEnemyNormal	Method Takes in parameters int x, int y, Player _player Returns void	If x and y passed is not same as player position then make normal enemy and add it to enemylist
SpawnEnemyHoming	Method Takes in parameters int x, int y, Player _player Returns void	If x and y passed is not same as player position then make Homing enemy and add it to enemylist
Update	override method, Takes in parameter GameTime gametime, Returns void	Uses similar technique to that done for movement in enemynormal. Here it just adds random expect when choosing what type of enemy to spawn. Also if isexpired is true for any enemy, then move it to removeEnemyList and them remove it from _enemyList
Draw	Method, Returns Bool	Tells all enemies present in draw to draw theselves

Table 16: <<Projectile> details (may need to add a field and property for point later on) (currently kepts oldmovetime,x,y here as it can be useful when adding other enemy behaviours later on (if need be))

Responsibility	Type Details	Notes
Projectile	Constructor, Takes in parameters `Vector2 position, Vector2 dimension, string address, Game game`	It is used entity to make itself. Also it is going to set _isRemoved as false,
HitEnemy	Method Return void	It is used to make _isRemoved into true
OldMoveTime	Property, returns double _oldMoveTime and sets double _oldMoveTime	It is used to get and set the oldMoveTime
Damage	Property, returns int _damage and sets int _damage	It is used to get and set the damage value of projectile
IsRemoved	Property, returns bool _isRemoved	It is used to know whether projectile is removed or not
Update	override method,	Will update itself using its parent

	Takes in parameter GameTime gametime, Returns void	class if _isRemoved is set to false. Also sets its position using speed and direction
Draw	override method, Returns void	Will draw itself using its parent class if _isRemoved is set to false.

Table 17: <<ProjectileManager> details

Responsibility	Type Details	Notes
ProjectileManager	Constructor, Takes in parameters `Game game`	It is used to make 2 lists:_projectileList, _removprojectile. And also stores game into _game
ProjectileList	Property, returns List<Projectile> _projectileList and sets List<Projectile> _projectileList	It is used to get and set projectilelist
NormalBullet	Method, Takes in parameter Player player Returns void	It is used make a projectilenormal using helper class, set its directions using player direction and add it to list of projectile
SlowBullet	Method, Takes in parameter Player player Returns void	It is used make a projectileSlow using helper class, set its directions using player direction and add it to list of projectile
Update	method, Takes in parameter GameTime gametime, Returns void	Will tell each projectile its list to update itself, but if a projectile has isRemvoed set to true, it wil add it to removeProjectileList. Also it will remove all projectiles present in removeProjectileList
Draw	method, Returns void	Will tell all projectiles in projectilelist to draw themselves

Table 18: <<ProjectileNormal> details

Responsibility	Type Details	Notes
ProjectileNormal	Constructor, Takes in parameters `Vector2 position, Vector2 dimension, string address, Game game`	It is used to make projectileNormal. Also its sets damage and speed of it using values preset for it in global

Table 19: <<ProjectileSlow> details

Responsibility	Type Details	Notes
ProjectileSlow	Constructor, Takes in parameters `Vector2 position, Vector2 dimension, string address, Game game`	It is used to make projectileSlow. Also its sets damage and speed of it using values preset for it in global

Table 20: <<CollisionDetector> details (can call player.points later on here to set the point system for player)

Responsibility	Type Details	Notes
CollisionDetector	Constructor,	It is used to create two responders: _enemyPLayerCollideResponder, _enemyProjectileCollideResponder (it's following a design pattern but am unable to recall what it is at the moment)
Detect	Method, Takes in parameters Player player, `List<Enemy> enemyList, List<Projectile> projectileList`	It is used to detect whether minimum collision distance (set in global) has been reached between either enemy and player or enemy and projectile and if so performs respective actions
NormalBullet	Method, Takes in parameter Player player Returns void	It is used make a projectilenormal using helper class, set its directions using player direction and add it to list of projectile
SlowBullet	Method, Takes in parameter Player player Returns void	It is used make a projectileSlow using helper class, set its directions using player direction and add it to list of projectile
Update	method, Takes in parameter GameTime gametime, Returns void	Will tell each projectile its list to update itself, but if a projectile has isRemvoed set to true, it wil add it to removeProjectileList. Also it will remove all projectiles present in removeProjectileList
Draw	method, Returns void	Will tell all projectiles in projectilelist to draw themselves

Table 21: <EnemyProjectileCollisionPresponder> details

Responsibility	Type Details	Notes
`EnemyProjectileCollisionResponder`	Constructor,	Does nothing Only kept it for good practise and ensure it doesn't go wrong in any way
`EnemyProjectileCollide`	Method, Takes in parameters Projectile p, Enemy e	It is used to make projectile call hitenemy method and enemy to call washit method whiole passing in projectile's damage as parameter to the washit method

Table 22: <<CollisionDetector> details (can call player.points later on here to set the point system for player)

Responsibility	Type Details	Notes
CollisionDetector	Constructor,	It is used to create two responders: _enemyPLayerCollideResponder, _enemyProjectileCollideResponder (it's following a design pattern but am unable to recall what it is at the moment)
Detect	Method, Takes in parameters Player player, `List<Enemy> enemyList, List<Projectile> projectileList`	It is used to detect whether minimum collision distance (set in global) has been reached between either enemy and player or enemy and projectile and if so performs respective actions
NormalBullet	Method, Takes in parameter Player player Returns void	It is used make a projectilenormal using helper class, set its directions using player direction and add it to list of projectile
SlowBullet	Method, Takes in parameter Player player Returns void	It is used make a projectileSlow using helper class, set its directions using player direction and add it to list of projectile
Update	method, Takes in parameter GameTime gametime, Returns void	Will tell each projectile its list to update itself, but if a projectile has isRemvoed set to true, it wil add it to removeProjectileList. Also it will remove all projectiles present in removeProjectileList
Draw	method, Returns void	Will tell all projectiles in projectilelist to draw themselves

Table 23: <EnemyPlayerCollisionPresponder> details

Responsibility	Type Details	Notes
`EnemyPlayerCollisionResponder`	Constructor,	Does nothing Only kept it for good practise and ensure it doesn't go wrong in any way
`EnemyPlayerCollide`	Method, Takes in parameters Player p, Enemy e	It is used to make player call hitEnemy method and enemy to call washit method whole passing in projectile's damage as parameter to the washit method. (currently set player collide damage very high to kill all enemy with 1 collide at the expense of 1 health)

Table 24: <<KeyboardController> details

Responsibility	Type Details	Notes
KeyboardController	Constructor, Takes in parameter Player player	It is used to set _player as player, make a new dictionary called _commandLibrary and add keys and commands to the dictionary It follows the strategy pattern and I am using it here to bind the keys to the commands
Update	method, Returns void	Will set _currentCommand as nullCommand, then it will get keyboardstate and store it as keyboard state. Then it will use the key to look though its command library and find the command and set it as _current command and execute it

Table 25: <<ICommands> <<interface>> details

Responsibility	Type Details	Notes
Execute	Method Returns void	It will be used to ensure all the classes that uses it will have a n Execute method that takes no parameter and returns void. (here is used it as part of strategy pattern and added all of the commands into a single dictionary)

Table 26: <<DownCommand> details

Responsibility	Type Details	Notes
DownCommand	Constructor Takes parameter Player player	It is used to set _player as player
Execute	Method Returns void	If player is not touching bottom of screen then tell player to GoDown (calibrated it with respect to player's dimensions)

Table 27: <<LeftCommand> details

Responsibility	Type Details	Notes
LeftCommand	Constructor Takes parameter Player player	It is used to set _player as player
Execute	Method Returns void	If player is not touching left of screen then tell player to GoLeft (calibrated it with respect to player's dimensions)

Table 28: <<NullCommand> details

Responsibility	Type Details	Notes
NullCommand	Constructor	Does nothing. Only used it as placeholder for current command in keyboardController
Execute	Method	Does nothing. Only used it as placeholder for current command in keyboardController

Table 29: <<QuitCommand> details

Responsibility	Type Details	Notes
QuitCommand	Constructor	Does nothing. Only kept it for good practise and to ensure it doesn't go wrong in any way
Execute	Method Returns void	Exits code

Table 30: <<RightCommand> details

Responsibility	Type Details	Notes
RightCommand	Constructor Takes parameter Player player	It is used to set _player as player
Execute	Method Returns void	If player is not touching right of screen then tell player to GoRight (calibrated it with respect to player's dimensions)

Table 31: <<UpCommand> details

Responsibility	Type Details	Notes
UpCommand	Constructor Takes parameter Player player	It is used to set _player as player
Execute	Method Returns void	If player is not touching top of screen then tell player to GoUp (calibrated it with respect to player's dimensions)

For Global, it have used to to basically keep track of all the constants I have used in my code:

(although both player and enemy height and width are current kept same, I am using different variables to store the current constants as I may edits the values later on to make the enemy bigger or smaller than the player)

```
public static SpriteBatch spriteBatch;
  public static int screenHeight = 800;
  public static int screenWidth = 800;

  public static int projectileHeight = 20;
  public static int projectileWidth = 20;
  public static int projectileRotationVelocity = 3;
  public static int projectileLinearVelocity = 4;
  public static double shootDelay = 0.1;

  public static int projectileSlowDamage = 2;
  public static int projectileSlowSpeed = 5;

  public static int projectileNormalDamage = 1;
  public static int projectileNormalSpeed = 10;

  public static int enemyHeight = 50;
  public static int enemyWidth = 50;
  public static double moveTimeLimit = 2
  public static double spawnTimeLimit = 5;
  public static float enemyNormalSpeed = 1;
  public static int enemyNormalHealth = 1;
  public static float enemyHomingSpeed = 0.005f;
  public static int enemyHomingHealth = 5;

  public static int playerHeight = 50;
  public static int playerWidth = 50;
  public static int playerSpeed = 2;
  public static int playerHealth = 5;
  public static int playerCollideDamage = 10;

  public static int objectTouchDistance = 50;
```

I didn't use any enums in my code. Instead I used lists and dictionaries which I have already mentioned in their specific classes.