Game idea :
Survival strategy game
The Goal is to live as long as possible
You can build fa farm for food camp for solder and a mine if you're close to a mountain , u can upgrade the range the damage and the health of the solder.
The zombies will start killing the solder before going to you, and it will calculate the best rout and the closest solder to attack.

To start the game you have to go to the main class and after losing restart by closing and running the program again.

The terminal act as message Tool for you to see if you're gathering stuff if you get damage ect …

the building class :

This class represents the building in the game.

It stores the type of building as a String.

It also stores a color, which can be used to visually represent the building in the game.

The constructor is used to create a building by setting its type and color when the object is made.

The two getter methods:

getType() returns the building's type.

getColor() returns the building's color.

This class acts as a simple data holder that other parts of the game can use to identify and draw buildings.

the GameWindow class :

Overall, this class handles the main game window and input focus setup so the game can run and respond to player actions.

It creates a JFrame titled "Zombie Survival".

The window size is set to 600×600, it can't be resized, and it closes the program when exited.

A GamePanel is created and added to the frame — this panel is where the game is drawn and updated.

the ZombieManager class :

This class manages all zombie behavior in the game.

It keeps a static list of zombies, so all zombies are tracked in one place.

spawnZombies creates zombies at random edges of the map, making them enter from outside the territory.

randomEdgeSpawn chooses one of the four map edges and returns a valid spawn position.

During the game update:

Each zombie looks for the nearest soldier using Manhattan distance.

If a soldier is nearby:

The zombie attacks if adjacent.

Otherwise, it moves toward the soldier.

If no soldiers exist:

Zombies move toward the base.

If adjacent to the base, they damage it, and the game ends if the base HP reaches zero.

After zombies act:

Soldiers attack zombies in range.

Dead zombies are removed from the zombie list.

Dead soldiers are removed from the soldier list.

Utility methods:

allZombiesDead() checks if the wave is cleared.

getZombies() returns the current zombie list.

the GamePanel class :

his class is the core of the game. It controls the map, player actions, UI, game rules, and the day/night cycle.

It creates a 16×16 tile map, draws tiles, buildings, the player, soldiers, and zombies.

The player moves with the keyboard during the day only, and each action consumes a limited number of turns.

The base (command center) is placed in the center and acts as the main interaction point.

Gameplay systems handled here:

Resource gathering (wood, rock, iron, food) by standing next to them .

Building placement (farm, mine, camp, wall) with resource costs and turn usage.

Soldier management: spawning soldiers at camps, selecting them, and upgrading their health, damage, or range.

Button UI logic that shows or hides actions based on player position, time of day, and available turns.

Day/Night cycle:

During the day, the player explores, builds, and gathers.

When turns run out and the player returns to base, night starts.

At night, zombies spawn and are controlled by ZombieManager, attacking soldiers or the base.

When all zombies are defeated, a new day begins with more turns and stronger waves.

Rendering:

The paintComponent method draws the HUD, tiles, buildings, units, and resources.

The game ends if the base HP reaches zero, showing a game-over message (death).

the Tile class :

This class represents a single tile on the game map.

Each tile stores its type (grass, water, forest, wall, etc.), whether it’s accessible, its color, and its grid position.

Tiles can also act as walls, which have their own health value.

Main behavior:

Walls start with 5 HP when created.

damageWall reduces wall health when attacked.

When wall HP reaches zero, the tile reverts back to grass, becomes accessible again, and changes color.

Getter methods allow other systems (movement, building, zombies) to:

Check the tile’s type,

See if it can be walked on,

Get its position and color.

the Soldier class :

This class represents the soldier unit that defends the base from zombies.

A soldier has a position, health, attack damage, attack range, and a level.

Soldiers start as melee units with low stats and can be improved over time.

Core behavior:

The upgrade method lets soldiers improve range, damage, or health if enough resources are available.

Each successful upgrade increases the soldier’s level, up to a maximum level.

Soldiers can attack zombies if they are within range, using Manhattan distance.

Soldiers can take damage from zombies and die when their HP reaches zero.

Utility methods:

Getters provide position, attack power, and current HP.

Helper methods increase stats when upgrades are applied.

isDead() is used to remove soldiers from the game when killed.

the Zombie class :

This class defines the zombie enemy unit and its behavior.

A zombie has a position, health, and attack damage.

Zombies are enemies that move one tile at a time.

Movement :

moveToward makes the zombie move closer to a target (soldier or base).

It checks all adjacent tiles and chooses the one that reduces distance to the target.

Zombies avoid walls if possible.

If all paths are blocked by walls, the zombie attacks a wall, damaging it instead of moving.

Combat behavior:

Zombies attack when they are adjacent to a target.

takeDamage reduces zombie health when soldiers attack.

When HP reaches zero, the zombie is considered dead and removed.

Utility methods:

Position and stat getters are used by rendering and combat systems.

getAdjacentTiles provides nearby tiles for movement and wall detection.

the Main class :

This is the entry point of the game.

The main method is where the program starts running.

It calls GameWindow.createWindow(), which sets up the game window and loads the game panel.

it simply starts the game by launching the main window.