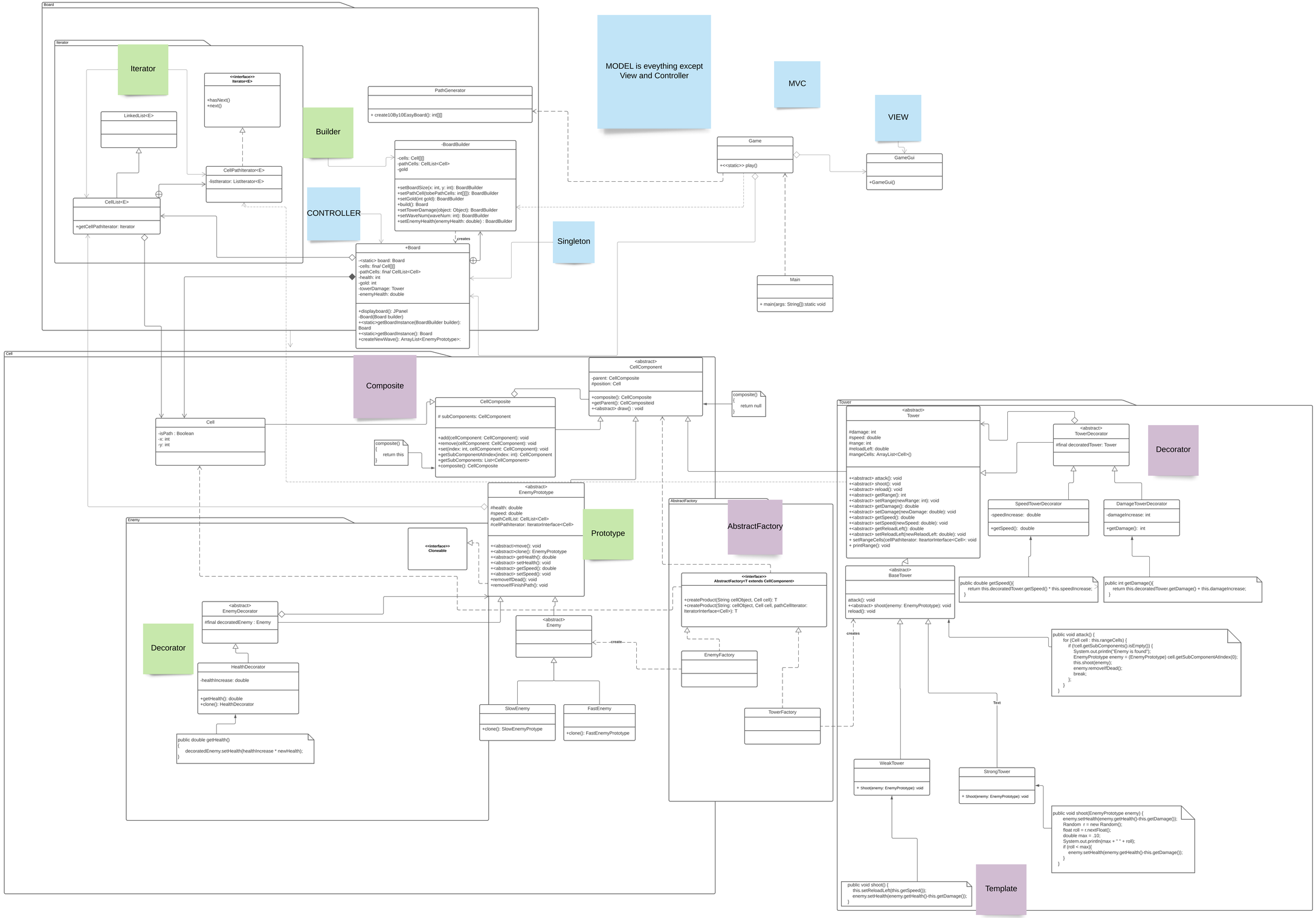
Board
Iterator
Iterator
<<interface>>
Iterator<E>
+hasNext()
+next()
LinkedList<E>
CellPathIterator<E>
-listIterator: ListIterator<E>
CellList<E>
+getCellPathIterator: Iterator
Builder
PathGenerator
+ create10By10EasyBoard(): int[][]
MODEL is eveything except View and Controller
MVC
VIEW
Game
+<<static>> play()
GameGui
+GameGui()
-BoardBuilder
-cells: Cell[][]
-pathCells: CellList<Cell>
-gold
+setBoardSize(x: int, y: int): BoardBuilder
+setPathCell(tobePathCells: int[][]): BoardBuilder
+setGold(int gold): BoardBuilder
+build(): Board
+setTowerDamage(object: Object): BoardBuilder
+setWaveNum(waveNum: int): BoardBuilder
+setEnemyHealth(enemyHealth: double) : BoardBuilder
CONTROLLER
creates
Singleton
+Board
-<static> board: Board
-cells: final Cell[][]
-pathCells: final CellList<Cell>
-health: int
-gold: int
-towerDamage: Tower
-enemyHealth: double
+displayboard(): JPanel
-Board(Board builder)
+<static>getBoardInstance(BoardBuilder builder): Board
+<static>getBoardInstance(): Board
+createNewWave(): ArrayList<EnemyPrototype>:
Main
+ main(args: String[]):static void
Cell
Composite
Cell
-isPath : Boolean
-x: int
-y: int
CellComposite
subComponents: CellComponent
+add(cellComponent: CellComponent): void
+remove(cellComponent: CellComponent): void
+set(index: int, cellComponent: CellComponent): void
+getSubComponentAtIndex(index: int): CellComponent
+getSubComponents: List<CellComponent>
+composite(): CellComposite
composite()
{
return this
}
<abstract>
CellComponent
-parent: CellComposite
#position: Cell
+composite(): CellComposite
+getParent(): CellCompositeid
+<abstract> draw() : void
composite()
{
return null
}
Tower
<abstract>
Tower
#damage: int
#speed: double
#range: int
#reloadLeft: double
#rangeCells: ArrayList<Cell>()
+<abstract> attack(): void
+<abstract> shoot(): void
+<abstract> reload(): void
+<abstract> getRange(): int
+<abstract> setRange(newRange: int): void
+<abstract> getDamage(): double
+<abstract> setDamage(newDamage: double): void
+<abstract> getSpeed(): double
+<abstract> setSpeed(newSpeed: double): void
+<abstract> getReloadLeft(): double
+<abstract> setReloadLeft(newReloadLeft: double): void
+ setRangeCells(cellPathIterator: IteartorInterface<Cell>: void
+ printRange(): void
<abstract>
TowerDecorator
#final decoratedTower: Tower
Decorator
SpeedTowerDecorator
-speedIncrease: double
+getSpeed(): double
DamageTowerDecorator
-damageIncrease: int
+getDamage(): int
public double getSpeed(){
return this.decoratedTower.getSpeed() * this.speedIncrease;
}
public int getDamage(){
return this.decoratedTower.getDamage() + this.damageIncrease;
}
<abstract>
BaseTower
attack(): void
+<abstract> shoot(enemy: EnemyPrototype): void
reload(): void
creates
public void attack() {
for (Cell cell : this.rangeCells) {
if (!cell.getSubComponents().isEmpty()) {
System.out.println("Enemy is found");
EnemyPrototype enemy = (EnemyPrototype) cell.getSubComponentAtIndex(0);
this.shoot(enemy);
enemy.removeIfDead();
break;
};
}
}
Text
WeakTower
+ Shoot(enemy: EnemyPrototype): void
StrongTower
+ Shoot(enemy: EnemyPrototype): void
public void shoot(EnemyPrototype enemy) {
enemy.setHealth(enemy.getHealth()-this.getDamage());
Random r = new Random();
float roll = r.nextFloat();
double max = .10;
System.out.println(max + " " + roll);
if (roll < max){
enemy.setHealth(enemy.getHealth()-this.getDamage());
}
}
public void shoot() {
this.setReloadLeft(this.getSpeed());
enemy.setHealth(enemy.getHealth()-this.getDamage());
}
Template
<abstract>
EnemyPrototype
#health: double
#speed: double
#pathCellList: CellList<Cell>
#cellPathIterator: IteratorInterface<Cell>
+<abstract>move(): void
+<abstract>clone(): EnemyPrototype
+<abstract> getHealth(): double
+<abstract> setHealth(): void
+<abstract> getSpeed(): double
+<abstract> setSpeed(): void
+removeIfDead(): void
+removeIfFinishPath(): void
Prototype
Enemy
<<interface>>
Cloneable
<abstract>
EnemyDecorator
#final decoratedEnemy : Enemy
Decorator
HealthDecorator
-healthIncrease: double
+getHealth(): double
+clone(): HealthDecorator
public double getHealth()
{
decoratedEnemy.setHealth(healthIncrease * newHealth);
}
<abstract>
Enemy
--create--
SlowEnemy
+clone(): SlowEnemyProtype
FastEnemy
+clone(): FastEnemyPrototype
AbstractFactory
AbstractFactory
<<interface>>
AbstractFactory<T extends CellComponent>
+createProduct(String cellObject, Cell cell): T
+createProduct(String: cellObject, Cell cell, pathCellIterator: IteratorInterface<Cell>): T
EnemyFactory
TowerFactory