Assignment 4

1.

Using ‘Metrics Reloaded’, methods with design complexity ‘iv(G)’ and cyclomatic complexity ‘v(G)’ close to or above 10 are good candidates for refactoring.
For classes, weighted method complexity ‘WMC’ above 30 indicate classes that should be refactored. Average operation ‘OCavg’ complexities above 4 do too.

Methods:

- Ghost.update(double) has a cyclomatic complexity of 13/11. (two refactors)
- Game.update(double) has a cyclomatic complexity of 10.
- MapParser.parseSquare(Board,char,int,int) has a cyclomatic complexity of 12.
- Direction.keyToDirection(KeyCode,PacMan) has a cyclomatic complexity of 7.
- WallSprite.draw has a cyclomatic complexity of 8.

Classes:

- Ghost has a weighted method complexity (WMC) of 47
- MapParser has an average operation complexity of 4.5
- Direction has an average operation complexity of 2.14
  (although not as high as 4+, its one of the highest OCavg’s)
- Wallsprite has an average operation complexity of 7
- LoginDao and RegisterDao had an average operation complexity of 3

2. Metrics:

Ghost.update(double) (1) & Game.update(double):

Before Refactor	After Refactor
Method,ev(G),iv(G),v(G)	Method,ev(G),iv(G),v(G)
pacman.logic.entity.Ghost.Mode.scatter(),2,1,2	pacman.logic.entity.Ghost.Mode.invertScatter(),2,1,2
pacman.logic.entity.Ghost.beScared(),1,1,1	pacman.logic.entity.Ghost.collideWithPacMan(PacMan),2,2,4
pacman.logic.entity.Ghost.justEaten(),1,1,1	pacman.logic.entity.Ghost.setEaten(),1,1,1
pacman.logic.entity.Ghost.unScare(),1,1,1	pacman.logic.entity.Ghost.setScared(boolean),1,1,2
pacman.logic.entity.Ghost.update(double),1,10,13	pacman.logic.entity.Ghost.update(double),1,6,9
pacman.logic.entity.PacMan.checkEatenGhosts(),1,4,4	pacman.logic.entity.PacMan.checkCollision(),2,2,2
pacman.logic.entity.PacMan.enterImmunity(),1,1,1	pacman.logic.entity.PacMan.collideWithPacMan(PacMan),1,1,1
pacman.logic.entity.PacMan.enterPumped(),2,4,4	pacman.logic.entity.PacMan.setImmunity(),1,1,1
pacman.logic.entity.PacMan.exitPumped(),1,1,1	pacman.logic.entity.PacMan.setPumped(boolean),1,2,2
pacman.logic.game.Game.update(double),2,9,10	pacman.logic.entity.PacMan.updateTimers(double),1,3,4
pacman.logic.level.Board.computeScore(),1,2,2	pacman.logic.entity.Pellet.collideWithPacMan(PacMan),1,1,1
pacman.logic.level.Level.eatPowerPellet(),1,1,1	pacman.logic.entity.PowerPellet.collideWithPacMan(PacMan),1,1,1
	pacman.logic.entity.Wall.collideWithPacMan(PacMan),1,1,1
	pacman.logic.game.Game.update(double),2,2,3
	pacman.logic.level.Board.addTickScore(int),1,1,1
	pacman.logic.level.Board.getTickScore(),1,1,1
	pacman.logic.level.Board.resetTickScore(),1,1,1
Class,OCavg,WMC	Class,OCavg,WMC
pacman.logic.entity.Ghost,"2,13",32	pacman.logic.entity.Ghost,"2,33",35
pacman.logic.entity.PacMan,"1,62",13	pacman.logic.entity.PacMan,"1,67",15
pacman.logic.entity.Pellet,"1,00",4	pacman.logic.entity.Pellet,"1,00",5
pacman.logic.entity.PowerPellet,"1,00",4	pacman.logic.entity.PowerPellet,"1,00",5
pacman.logic.entity.Wall,"1,00",4	pacman.logic.entity.Wall,"1,00",5
pacman.logic.game.Game,"1,92",23	pacman.logic.game.Game,"1,50",18
pacman.logic.level.Board,"1,50",24	pacman.logic.level.Board,"1,44",26
pacman.logic.level.Level,"1,00",7	pacman.logic.level.Level,"1,00",6
Package,v(G)avg,v(G)tot	Package,v(G)avg,v(G)tot
pacman.logic.entity,"1,88",122	pacman.logic.entity,"1,81",125
pacman.logic.game,"1,74",47	pacman.logic.game,"1,48",40
pacman.logic.level,"2,15",101	pacman.logic.level,"2,10",101
Module,v(G)avg,v(G)tot	Module,v(G)avg,v(G)tot

template.main,"1,78",441	template.main,"1,73",437
Project,v(G)avg,v(G)tot	Project,v(G)avg,v(G)tot
project,"1,78",441	project,"1,73",437

Ghost.update(double) (2):

Before	After
Method,ev(G),iv(G),v(G)	Method,ev(G),iv(G),v(G)
pacman.logic.entity.Ghost.update(double),1,8,11	pacman.logic.entity.Ghost.update(double),1,4,4
	pacman.logic.entity.Ghost.updateChoice(),3,3,4
	pacman.logic.entity.Ghost.updateTimers(double),1,2,5
Class,OCavg,WMC	Class,OCavg,WMC
pacman.logic.entity.Ghost,"2,76",47	pacman.logic.entity.Ghost,"2,58",49
Package,v(G)avg,v(G)tot	Package,v(G)avg,v(G)tot
pacman.logic.entity,"1,80",148	pacman.logic.entity,"1,79",150
Module,v(G)avg,v(G)tot	Module,v(G)avg,v(G)tot
template.main,"1,78",441	template.main,"1,73",437
Project,v(G)avg,v(G)tot	Project,v(G)avg,v(G)tot
project,"1,77",558	project,"1,77",560

Ghost

Before	After
Method,ev(G),iv(G),v(G)	Method,ev(G),iv(G),v(G)
pacman.logic.entity.Ghost.breadthFirstSearch(Square, List)4,6,6	pacman.logic.level.Square.breadthFirstSearch(Square,List),4,6,6
pacman.logic.entity.Ghost.manhattenDistance(Square, List),1,2,3	pacman.logic.level.Square.manhattenDistance(Square,List),1,2,3
Class,OCavg,WMC	Class,OCavg,WMC
pacman.logic.entity.Ghost,"2,48",49	pacman.logic.entity.Ghost,"2,41",41
pacman.logic.level.Square,"1,79",25	pacman.logic.level.Square,"2,06",33
Package,v(G)avg,v(G)tot	Package,v(G)avg,v(G)tot
pacman.logic.entity,"1,79",150	pacman.logic.entity,"1,72",141
pacman.logic.level,"2,16",110	pacman.logic.level,"2,25",119
Module,v(G)avg,v(G)tot	Module,v(G)avg,v(G)tot
Project,v(G)avg,v(G)tot	Project,v(G)avg,v(G)tot

MapParser.parseSquare(Board,char,int,int) & MapParser

Before	After
Method,ev(G),iv(G),v(G)	Method,ev(G),iv(G),v(G)
pacman.logic.level.MapParser.parseSquare(Board,char,int,int),2,2,12	pacman.logic.level.MapParser.parseSquare(Board,char,int,int),2,1,4
Class,OCavg,WMC	Class,OCavg,WMC
pacman.logic.level.MapParser,"4,50",27	pacman.logic.level.MapParser,"2,83",17
Package,v(G)avg,v(G)tot	Package,v(G)avg,v(G)tot
pacman.logic.level,"2,25",119	pacman.logic.level,"2,09",111
Module,v(G)avg,v(G)tot	Module,v(G)avg,v(G)tot
template.main,"1,75",555	template.main,"1,73",547
Project,v(G)avg,v(G)tot	Project,v(G)avg,v(G)tot
project,"1,75",555	project,"1,73",547

Direction.keyToDirection(KeyCode,PacMan) & Direction

Before	After
Method,ev(G),iv(G),v(G)	Method,ev(G),iv(G),v(G)
pacman.logic.Direction.keyToDirection(KeyCode,PacMan),2,3,7	pacman.logic.Direction.keyToDirection(KeyCode,PacMan),1,2,2
Class,OCavg,WMC	Class,OCavg,WMC
pacman.logic.Direction,"2,14",15	pacman.logic.Direction,"1,43",10
Package,v(G)avg,v(G)tot	Package,v(G)avg,v(G)tot
pacman.logic,"1,60",24	pacman.logic,"1,27",19
Module,v(G)avg,v(G)tot	Module,v(G)avg,v(G)tot
template.main,"1,77",560	template.main,"1,75",555
Project,v(G)avg,v(G)tot	Project,v(G)avg,v(G)tot
project,"1,77",560	project,"1,77",560

WallSprite.draw & WallSprite

Before	After
Method,ev(G),iv(G),v(G)	Method,ev(G),iv(G),v(G)
pacman.graphics.sprite.WallSprite.draw(Wall,GraphicsContext,Style,double),1,8,8	pacman.graphics.sprite.WallSprite.draw(Wall,GraphicsContext,Style,double),1,4,4
	pacman.graphics.sprite.WallSprite.drawSide(Wall,GraphicsContext,Style,Direction)1,5,5
Class,OCavg,WMC	Class,OCavg,WMC
pacman.graphics.sprite.WallSprite,"7,00",7	pacman.graphics.sprite.WallSprite,"4,00",8
pacman.logic.level.Square,'2,06",33	pacman.logic.level.Square,"2,00",34
Package,v(G)avg,v(G)tot	Package,v(G)avg,v(G)tot
pacman.graphics.sprite,"1,52",44	pacman.graphics.sprite,"1,50",45
pacman.logic.level,"2,09",111	pacman.logic.level,"2,07",112
Module,v(G)avg,v(G)tot	Module,v(G)avg,v(G)tot
template.main,"1,73",547	template.main,"1,72",549

Project,v(G)avg,v(G)tot	Project,v(G)avg,v(G)tot
project,"1,73",547	project,"1,73",547

LoginDao.attempt(User) & RegisterDao.addUser(User)

Before	After
Method,ev(G),iv(G),v(G)	Method,ev(G),iv(G),v(G)
pacman.database.LoginDao.attempt(User),2,9,14	pacman.database.LoginDao.attempt (User),2,9,7
	pacman.database.LoginDao.executeQuery (User),2,9,7
Class,OCavg,WMC	Class,OCavg,WMC
pacman.database.LoginDao(User),3,16,13	pacman.database.LoginDao(User),3,16,9
Package,v(G)avg,v(G)tot	Package,v(G)avg,v(G)tot
pacman.database"1,68",52	pacman.graphics.sprite,"1,60",45
Module,v(G)avg,v(G)tot	Module,v(G)avg,v(G)tot
template.main,"1,79",647	template.main,"1,78",630
Project,v(G)avg,v(G)tot	Project,v(G)avg,v(G)tot
project,"1,79",647	project,"1,78",630

Refactor Operation:

Methods:

Ghost.update(double):

(1): collisions with pacman have been moved to dedicated abstract entity.collideWithPacMan(PacMan) function, called by pacman.update(double). This as until now, all this logic was indiscriminately placed and thus increased complexity of the code, both numerically and from a design perspective.

A minor change to keep PacMan's complexity down was moving timer logic to a seperate method. This happened in increments and is thus hard to measure the metrics of, which were not included for this reason.

(2): The code for a timer was moved to a updateTimer method, which was made into an abstract method of MovingEntity, as Pacman also uses a method by this name.

The code for updating a ghost's choice was also moved to a seperate updateChoice method. This made the ghost's update method less complex and much more readable and logical.

These changes reduced the cyclomatic complexity from 13 to 4.

Game.update(double):
Due to collision and timer handling and the game's update method was very complex, handling logic that belonged to entities themselves for both timing and scoring points. The timing was moved to the classes those timers belonged to. And with the help of the previously mentioned collideWithPacMan function, the score logic could be removed from the Game.update method, it being replaced by an integer the game keeps track of, namely the 'tick score', which is increment during a game cycle (tick) by for example pellets when they *collide with pacman*.
These changes reduced the cyclomatic complexity from 10 to 3.

MapParser.parseSquare(Board,char,int,int)
The method had its switch statement removed and replaced with a static hashmap, using a char as key and class as value. This together with the slightly unknown .getConstructor and .newInstance methods.
These changes reduced the cyclomatic complexity from 12 to 4.

Direction.keyToDirection(KeyCode,PacMan):
The method initially made use of a switch case which added a direction to each of the keys clicked. Using this method, the cyclomatic complexity was 7. however, after changing it to make use of a static hashmap, using a KeyCode as the key and Direction as the value, the cyclomatic complexity dropped to 2.

WallSprite.draw:
The splitting up of the method by making a private drawSide method, purely for drawing specific sides of the wall, together with giving Square a method for returning the manhatten distance between 2 squares, instead of finding the closest option, reduced the cyclomatic complexity from 8 to 4, although the new method has 5.

LoginDao.attempt(User):
Method checked the authenticity of the user that is trying to login with the provided credentials by using an if statement to check for each one of the user's credentials to be the correct ones, having such an cyclomatic complexity of 5 (by verifying username, password and score).It was extracted from the structure of the login db instance and was inserted into a separate method, which did not look for score equality anymore as username and password should suffice.The complexity was reduced down to 3.

Classes:

Ghost:

The manhattenDistance and breadthFirstSearch methods were moved to Square, as these logically belong more to this class than to Ghost itself.
Due to limited changes made in these methods this lead to a division of WMC between Ghost and Square, thus total complexity wasn't lowered much.
Sadly it was not possible to lower both WMC's a lot further without very large amounts of restructuring. This being the case we settled for this refactor.
These changes reduced the Weighted Method Complexity from 47 to 41.

MapParser:

See MapParser.parseSquare(Board,char,int,int) above.
These changes reduced the average operation complexity from 4.5 to 2.84

Direction:

See Direction.keyToDirection(KeyCode,PacMan) above.
These changes reduced the average operation complexity from 2.14 to 1.43.
It also reduced the Weighted Method Complexity from 15 to 10.

WallSprite:

See Wallsprite.draw above.
These changed reduced the average operation complexity from 7.00 to 4.00.

LoginDao:

See LoginDao.attempt (User) above.
The improvement was made by reducing cyclomatic complexity from 5 to 3.