Team Details

Carl White	Student Number: 12406138	CS Login: SE314065
Keith White	Student Number: 12406092	CS Login: SE314064
Gytis Stankevicius	Student Number: 12519977	CS Login: SE314062

Exercise 2.1

Noun and Verb Approach

Design a **software system** to **control** a new **automatic teller machine (ATM)** with the **following** requirements : **Users** of the **ATM** will be able to **access** their **account** **using** their **ATM card** and **PIN number**. **Account Holders** will **get** **three attempts** to **enter** the correct **pin number**, after which the **machine** will **retain** the **card**, notify **bank officials** and **freeze** the **customers account**. If the **customer** successfully **enters** the **correct pin number** associated with the **bank card**, depending upon the type of **account** they will **get** a **number** **of options**. Savings **account holders** may deposit **money** into their **account** and **withdraw** a designated **amount of money** once per **month**. **Current account holders** may **deposit** money into their **accounts**, **transfer** **money** to their **savings account**, **withdraw** **cash** up to a daily **limit** of 700 euro, and **check** the last 10 **transactions** on their **account**. All **account holders** may **check** their **balance**.

- Nouns are in **blue** and verbs in **green**

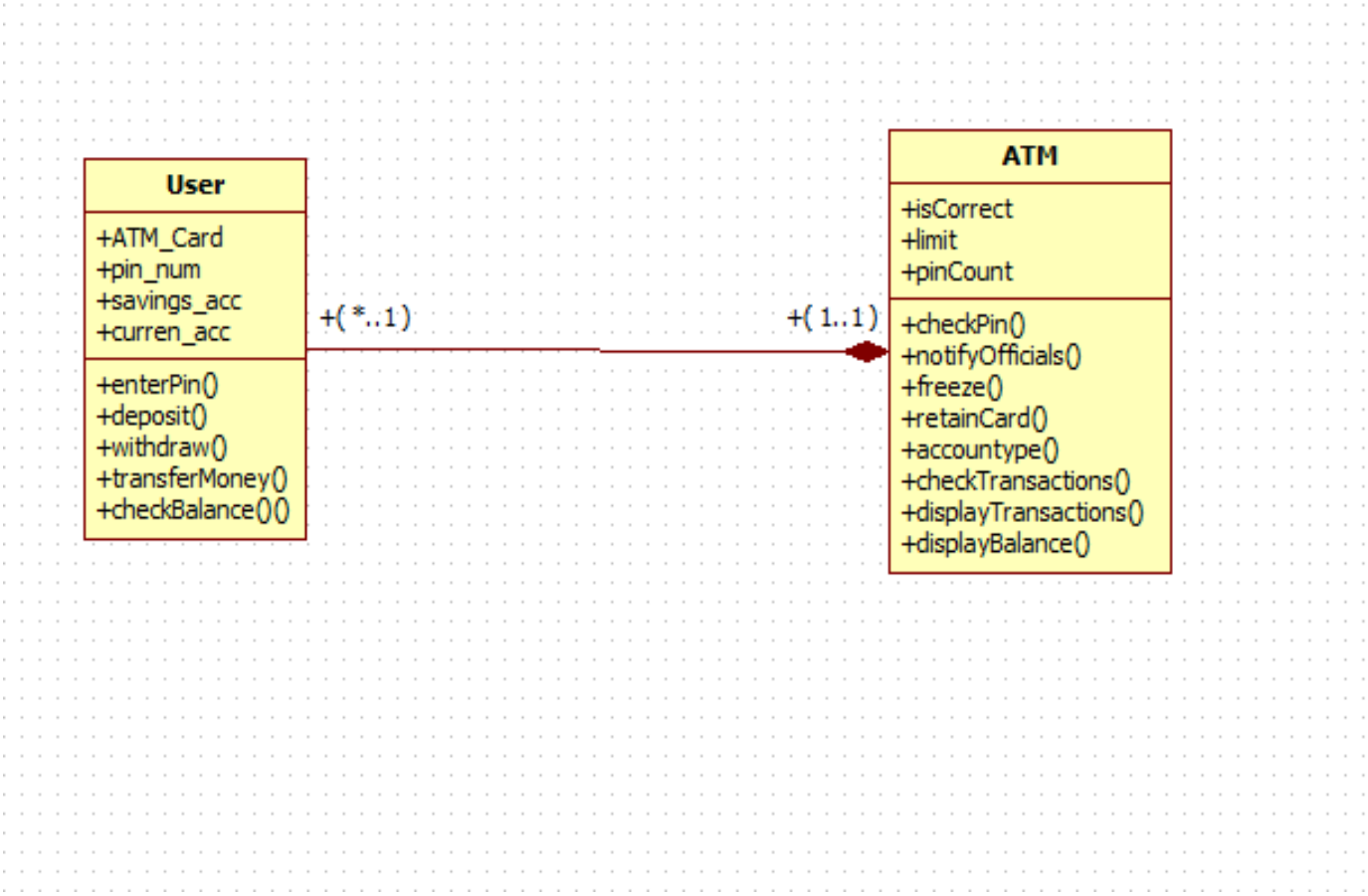
candidate classes: Removing redundancies such as Users, Account Holders and Customers we are left with:
Software system, ATM, User, Account, Money, Bank officials, Bank card, Pin number, Balance and Limit.

critical/major classes and why ?:

The two major classes are User and ATM, Items such as account are more suited to subclasses of the base User. Money, balance pin number and Limit should be member variables.

User:
Uses the ATM and has ATM card and Pin .Access account using ATM card and Pin.
3Attempts to enter pin correctly (pinCount)
Savings Account:
deposit/withdraw once per month/check balance
Current Account:
deposit/transfer to savings/withdraw limit/check last 10 transactions/check balance
ATM:
checks the Pin/retains card if incorrect 3 times
accountType.

displays transactions/withdraws money / display balance / notify officials



Exercise 2.2

Noun and Verb Approach

You have been asked to design a **multiplayer asteroids game**. The game environment will consist of three **game objects**: **spaceships, laser pulses, and asteroids**. All objects will have a **2D position, orientation, and velocity**.
The game will consist of between **1-4 players** each of which is associated with a separate spaceship. At the beginning of the game a collection of **asteroids** will be **created** with **random positions, orientations, velocities, and sizes**. During operation of the game, when the **user** **fires** their ship's cannon it will **emit** a **laser pulse**. Each laser pulse will have a **fixed velocity** and an **associated player** (i.e. the player associated with the ship that fired the laser). Each **player** will be identified by a **username** which will be **input** at the beginning of the game. For each game, each **player** will **start** with **3 lives**. **Players** will also **maintain** a **score** which will **start** at **zero** and can be **updated** by the **game system** as the **game progress**.
In games of less than 4 human players **additional computer controlled players** will be **added** in order to bring the total number of players to 4. At anytime each player can be queried for their next move. The **next move** of human players will be **determined** by **polling** their associated controller for its current state (e.g. which buttons are pressed). **Computer controlled players** on the other hand **will have** an associated AI engine which will **provide** a think method that will make the decision on the next move. Three levels of AI will be **provided**: **naive, medium, and advanced.**
The **operation of the game** will be **driven** by a game loop that **updates** the system state 60 times per second. To do this the system calls will **request** that each **asteroid, laser pulse, and spaceship** **updates** itself. **Asteroids** and **laser pulses** can be updated using standard physics models. A **spaceships** update method computes the new position by first requesting the player's next move.
After each iteration of the game loop a **collision detection system** is **employed** to **detect collisions** between objects. If a **collision** **occurs** between a **spaceship** and an **asteroid**, the **collision detection system** **kills** the **player**, thereby **deducting** a **life**. If the **player** still **has** **lives remaining** the **position of the spaceship** is **set** to a **random position of freespace**. If a **collision occurs** between a **laser pulse** and an **asteroid** the system **deletes** the **laser pulse** and **increments** the **score** of the **player** associated with the laser pulse. The **collision detection system** also **reduces** the **size** of the current asteroid to half its current size or **deletes** the **asteroid** if it has **reached** a **minimum size**.

Irrelevant Items:

> 1-4 players/beginner asteroids with random positions, orientations, velocities, size/players begin with 3 lives/updates 60 frames per-second/

candidate classes:

SpaceShip
Laser pulse
Asteroid
Game

critical/major classes and why?;

SpaceShip:

position/orientation/velocity
update()/ isFired()/ Collision()/ requestMove()

Laser Pulse:

position/orientation/velocity/
update()/collision ()/ delete() / incrementScore()

Asteroids:

position/orientation/velocity/ size
update ()/collision ()/delete()/ reduceSize()

Game Engine:

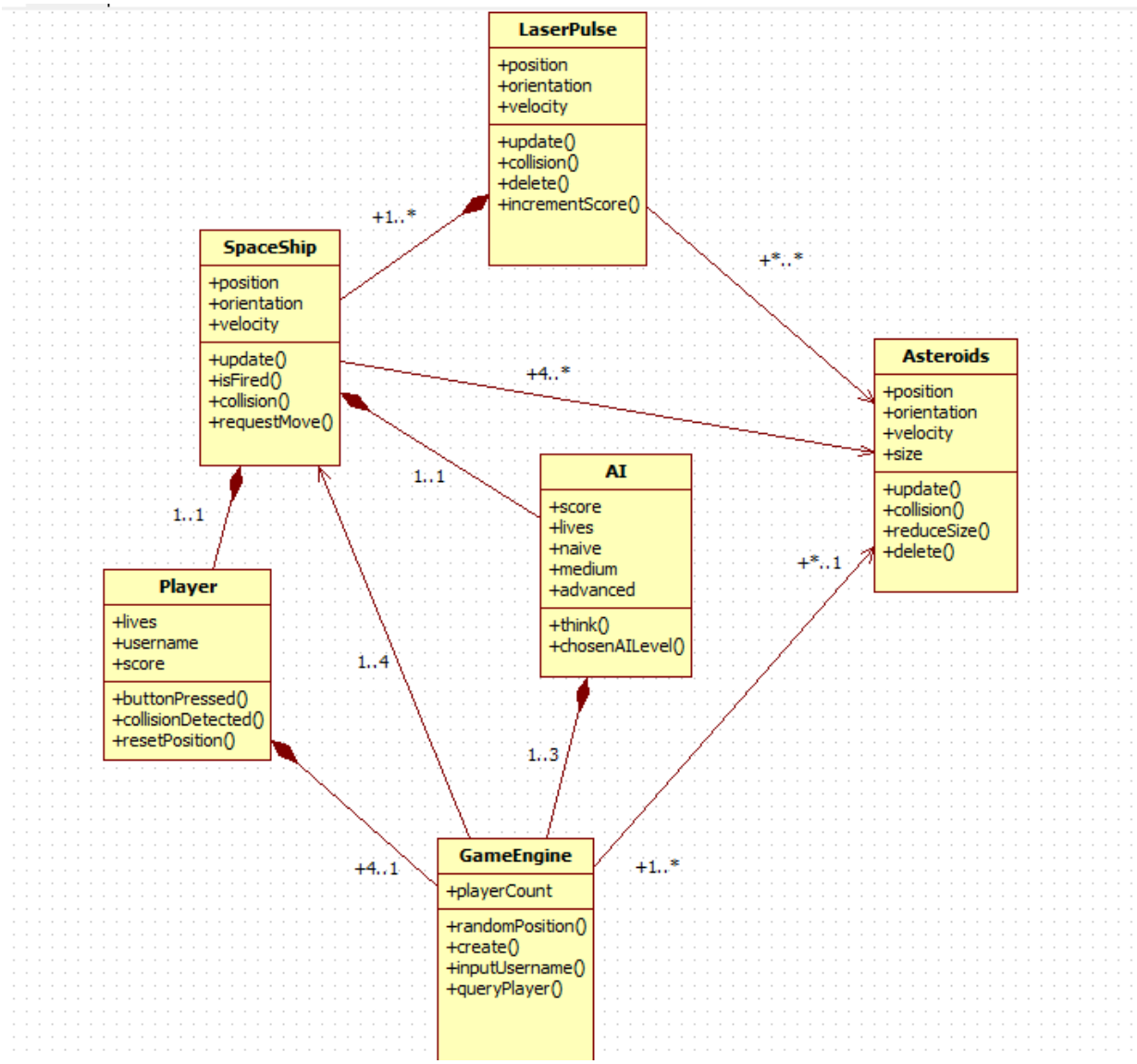
create ()/inputUsername()/
randomPosition()
playerCount

Player:

Lives, username, score
Buttonpressed(), collisionDetected(), resetPosition()

AI

Lives/ score/ naive/ medium/ advanced
Think()/ chooseAILevel()


LaserPulse
+position
+orientation
+velocity
+update()
+collision()
+delete()
+incrementScore()
SpaceShip
+position
+orientation
+velocity
+update()
+isFired()
+collision()
+requestMove()
Asteroids
+position
+orientation
+velocity
+size
+update()
+collision()
+reduceSize()
+delete()
AI
+score
+lives
+naive
+medium
+advanced
+think()
+chosenAILevel()
Player
+lives
+username
+score
+buttonPressed()
+collisionDetected()
+resetPosition()
GameEngine
+playerCount
+randomPosition()
+create()
+inputUsername()
+queryPlayer()
+1..*
+*..*
+4..*
1..1
1..1
+*..1
1..4
1..3
+4..1
+1..*