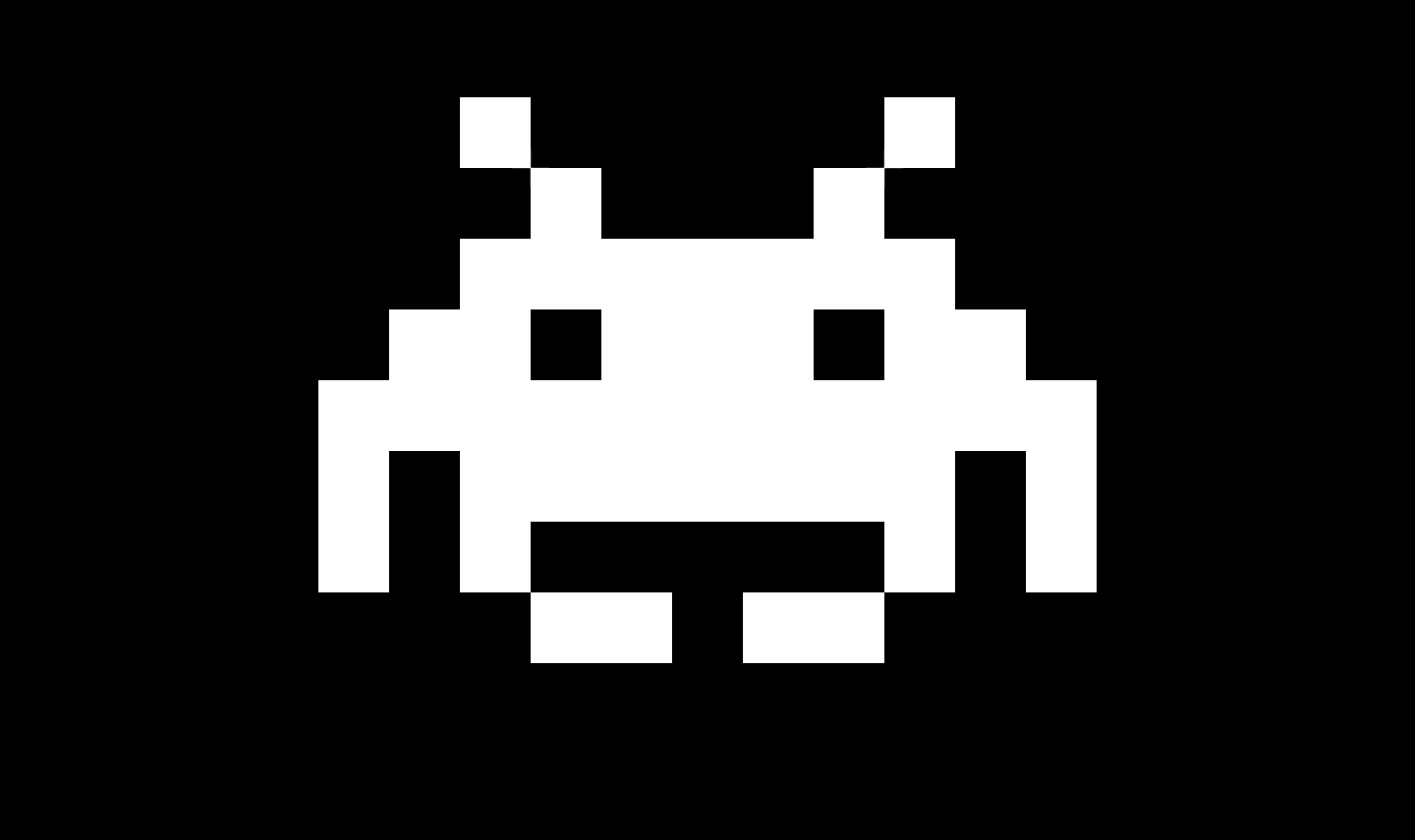
2020

Arlo Steyn

Grade 12 E

8/12/2020

IT PAT Phase 4

[](https://www.google.co.za/url?sa=i&url=https%3A%2F%2Fwww.destructoid.com%2Ftaito-and-degica-are-teasing-a-space-invaders-game-for-steam-478638.phtml&psig=AOvVaw3mkJjJR1h6JInQ3mTSj5hv&ust=1586792050925000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCKCsqvCb4-gCFQAAAAAdAAAAABAL)

Space Attack

PAT Phase 4

Technical and testing documentation

**Table of content**

**Technical documentation**

Externally sourced code…………………………………………………………………2

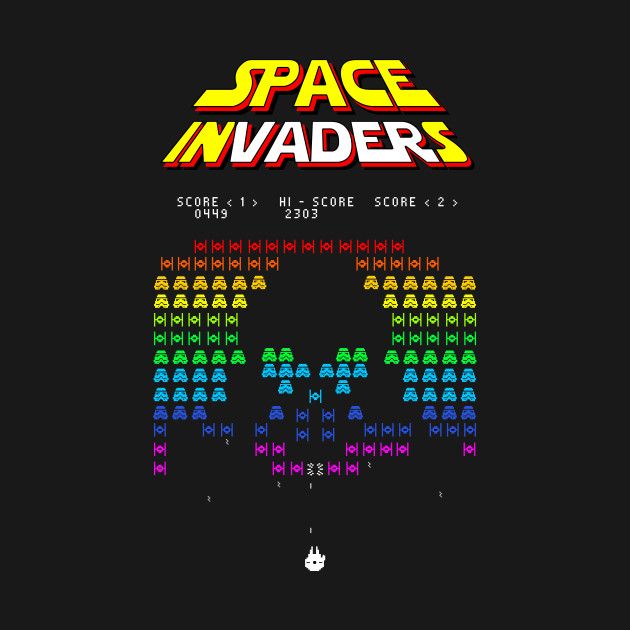
Explanation of critical algorithms……………………………………………………….4

Advanced techniques……………………………………………………………………8

**Testing documentation**

Test plan and results……………………………………………………………………11

Tested data and results…………………………………………………………………11



**Technical documentation**

**Externally sourced code**

**Timer**

TimerTask task = new TimerTask() {

public void run() {

i++;

Time = getTime(i);

//time = i;

Score++;

}

};

public void runTimer() {

//The void runTimer() method is used to schedule the timer every 1000

//miliseconds or 1 second

timer.schedule(task, 0, 1000);

}

This method was used to schedule a timer every 1 second.

<https://www.geeksforgeeks.org/java-util-timer-class-java/> (website copied from)

**ArrayLists**

import java.util.ArrayList;

private List<Alien> aliens;

aliens = new ArrayList<>();

aliens.add(alien);

An array list is an array that can have an infinite number of instances without the need of declaring the size of the array. Space Attack is a game that in theory can last forever depending on the powerups and skill level of the player, we therefore need the array list as the amount of alien objects per game is not known or finite.

<https://stackoverflow.com/questions/12340175/arraylist-when-making-a-game>

**Music**

private static void playMusic() {

try {

String songFile = new Music().getSong();

File musicFile = new File(songFile);

clip = AudioSystem.getClip();

clip.open(AudioSystem.getAudioInputStream(musicFile));

clip.start();

clip.loop(Clip.LOOP\_CONTINUOUSLY);

FloatControl gainControl = (FloatControl) clip

.getControl(FloatControl.Type.MASTER\_GAIN);

double gain = .3D;

float dB = (float) (Math.log(gain) / Math.log(10.0) \* 20.0);

gainControl.setValue(dB);

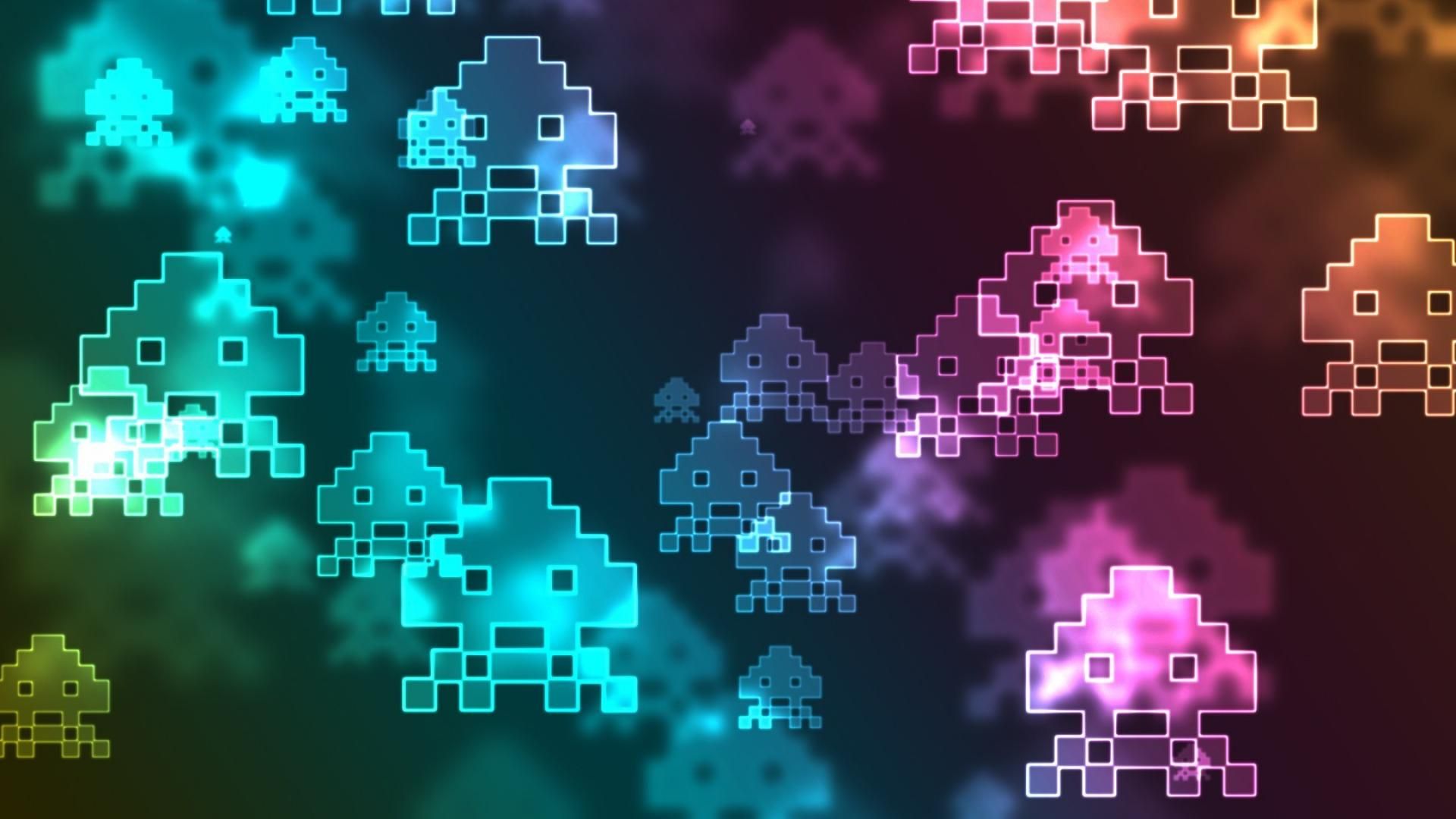
} catch (Exception i) {

i.printStackTrace();

}

}

<https://www.geeksforgeeks.org/play-audio-file-using-java/>(Copied and altered)

Music is needed in Space Attack to add to the theme (retro gaming)

**Explanation of critical algorithms**

**The “GameChange” Algorithm**

public void GameChange() {

if (alienCount < 1) {

change++;

if (change % 2 == 0 && change % 4 != 0 && !(change > 23)) {

addRows++;

}

if (change % 3 == 0 && change % 6 != 0 && change != 15 && change != 27 && !(change > 34)) {

addColumns++;

}

if (change % 4 == 0) {

alienBulletspeedChange++;

}

if (change % 6 == 0) {

speedChange++;

}

if (change == 7 || change == 13 || change == 19 || change == 25 || change == 33 || change == 37) {

speedChange = 1;

}

for (int i = 0; i < addRows; i++) {

for (int j = 0; j < addColumns; j++) {

Alien alien = new Alien(200 + 55 \* j, 10 + 40 \* i, imgArr[i]);

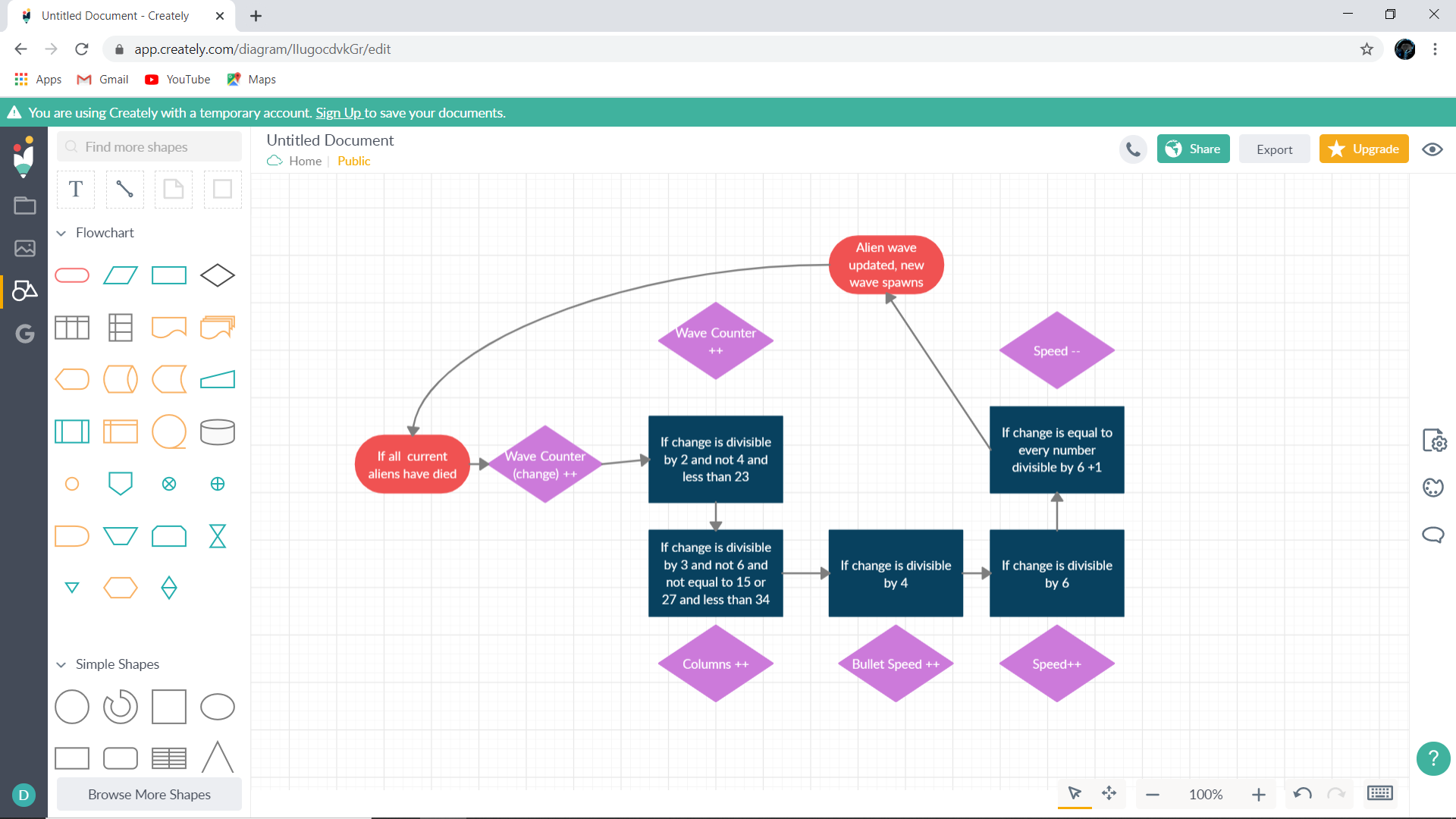
aliens.add(alien);

alienCount++;

}

} } }

Space Attack is a game that get progressively more difficult as the user clears/destroys the various waves of aliens. The game starts off with 1 row of 6 (columns) aliens. Naturally after each wave of aliens is cleared, the game should get more difficult, otherwise the game will be too repetitive and unenjoyable. To make the game progressively harder, the “GameChange” algorithm has been implemented. This method works in a “wave” concept (waves of enemies). Each wave is equated to the number of new aliens spawned whenever the previous wave has been cleared. “change” is the variable responsible for counting the amount of waves and is incremented whenever all the aliens on the screen are currently cleared (when the “alienCount” variable is equal to 0). The aliens’ speed, number of rows, columns and bullet speed are all stored in a variable format as seen above, we can thus change/alter these variables depending on what wave the aliens are on. As seen in the algorithm, if the wave counter (change) is divisible by 2 and not 4, the number of alien rows are increased by 1 until a maximum of 7 rows are on screen. The same concept applies for the columns of the aliens, if the wave counter is divisible by 3, not 6, not equal to 15/27 and it is less than 34, more alien columns are added, this eventually allows only a maximum of 10 columns of aliens. The same concept applies for the alien bullet/speed increase (The rows and columns are maxed because the game still needs to be playable after reaching a certain wave, therefore the bullet speed and speed of the increase is not capped, thus allowing the game to still get harder after the waves of the aliens’ columns and row increases end). After all the current aliens die on screen, a new one spawns with altered stats as seen above.

Flow chart representing how the algorithm works.

**Collision detection and Aliens shooting back**

public void AlienShootandDetection() {

Random generator = new Random();

for (Alien alien : aliens) {

int shot = generator.nextInt(500);

AlienBullet AB = alien.getAlienBullet();

if ((shot == 10 | shot == 20) && alien.isVisible() && AB.isDestroyed()) {

AB.setDestroyed(false);

AB.setX(alien.getX() + 20);

AB.setY(alien.getY() + 20);

}

if (!AB.isDestroyed()) {

AB.setY(AB.getY() + 10 + alienBulletspeedChange);

}

int AlienBulletX = AB.getX();

int AlienBulletY = AB.getY();

int playerX = player.getX();

int playerY = player.getY();

if (player.isVisible() && !AB.isDestroyed()) {

if (AlienBulletX >= (playerX) && AlienBulletX <= (playerX + 75)

&& AlienBulletY >= (playerY) && AlienBulletY <= (playerY + 75)) {

AB.setDestroyed(true);

Lives--;

}

}

if (!AB.isDestroyed()) {

AB.setY(AB.getY() + 1);

if (AB.getY() >= 700) {

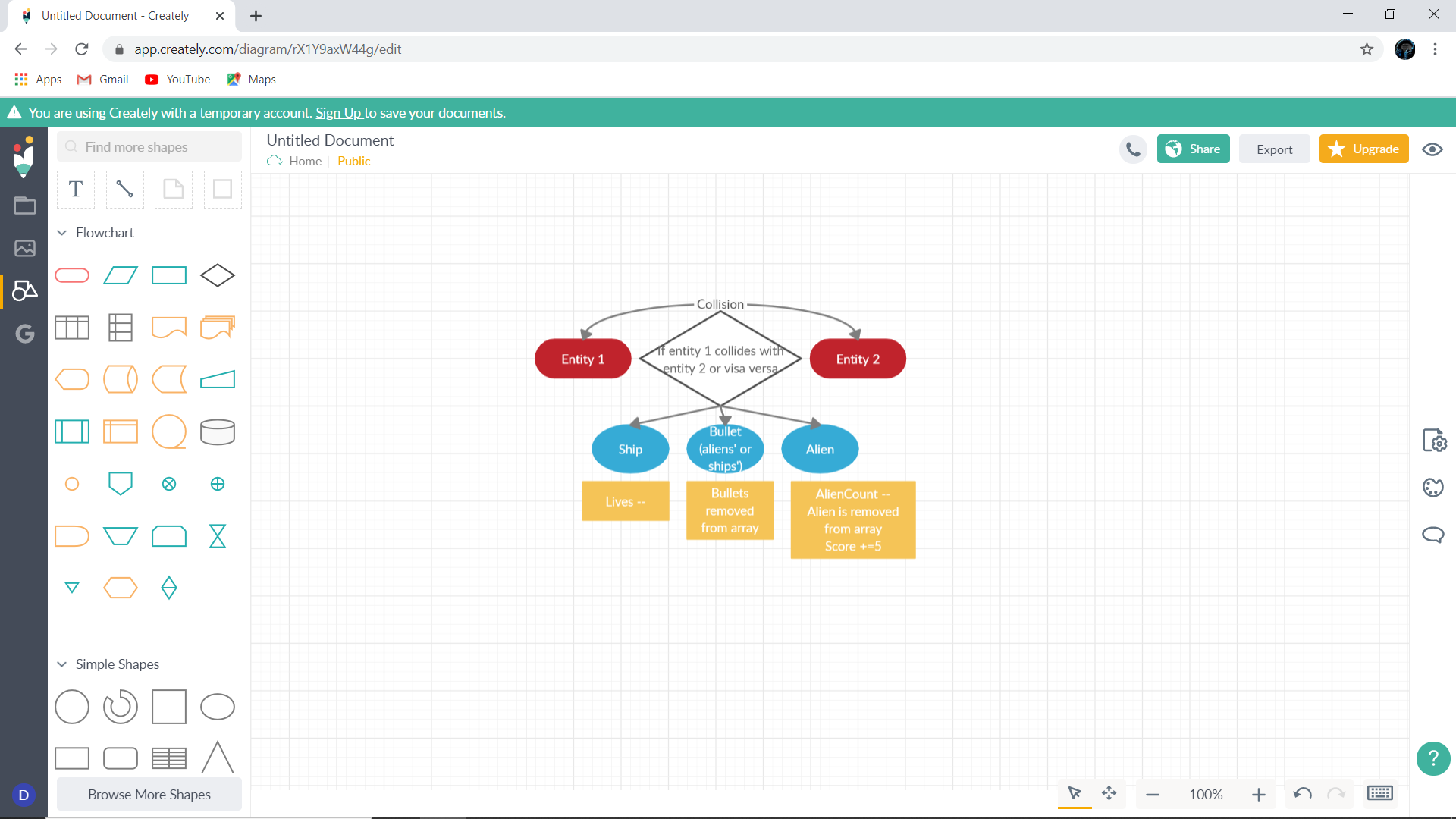
AB.setDestroyed(true);

}}}}

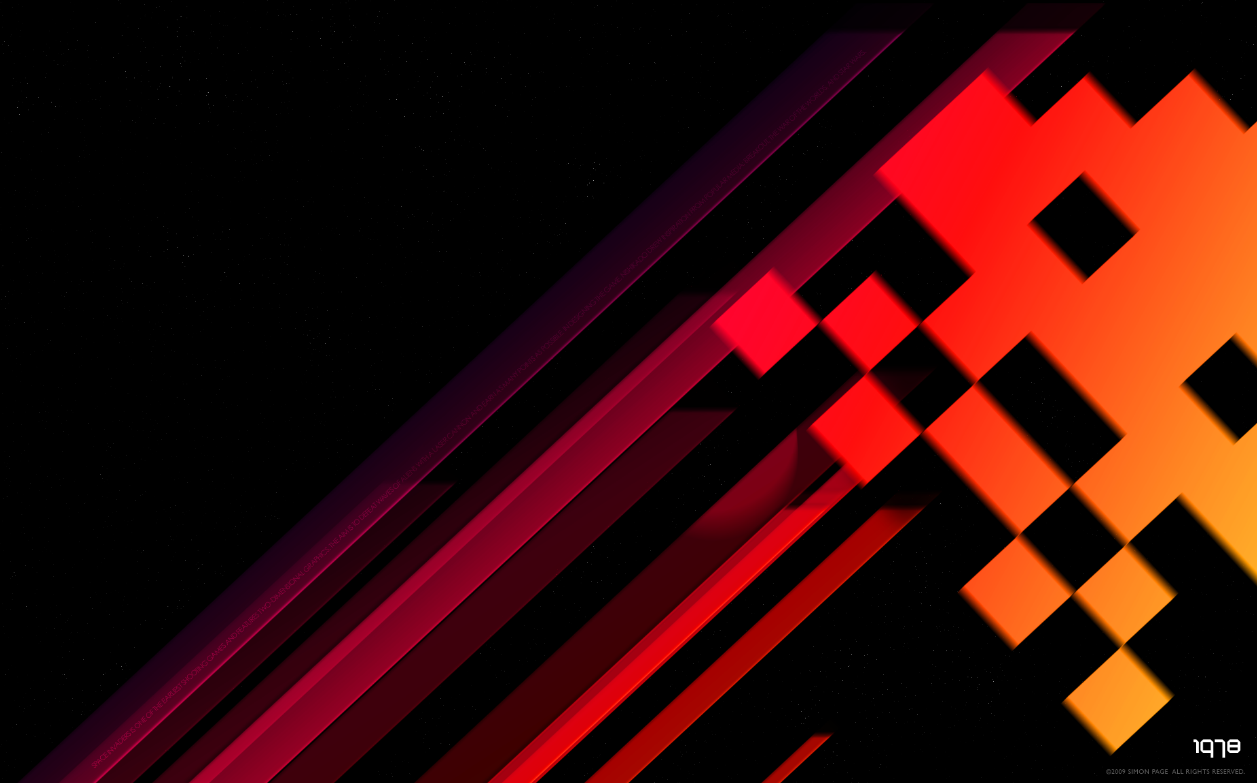
The algorithm contains both bullets hit detection as well as the ability for the aliens to randomly shoot. “ShootAlienDetection”, “AlienShipDetection” and the method above also contains collision detection between the various entities in the game. The above collision detection method will be used to represent the collision detection algorithms for the other methods.

The collision works as follows, if an entity collides with another entity (Ship, bullet or alien), an action is performed. These actions include destroying the entity/removing it from the array, increasing a score variable, deducting live points etc.

Flow chart representing collision detection.



Aliens ability to shoot:

Every 40 milliseconds (timer that repaints screen, performs actions etc) 500 random numbers are generated. If these numbers are equal to either 10 or 20, a random alien is chosen and given an AlienBullet object that shoots a projectile.

**Advanced techniques**

**PowerUps**

PowerUps are randomly chosen out of 6 every 30 seconds and can be used by the user by pressing the “P” key on the keyboard. The following powerups were added:

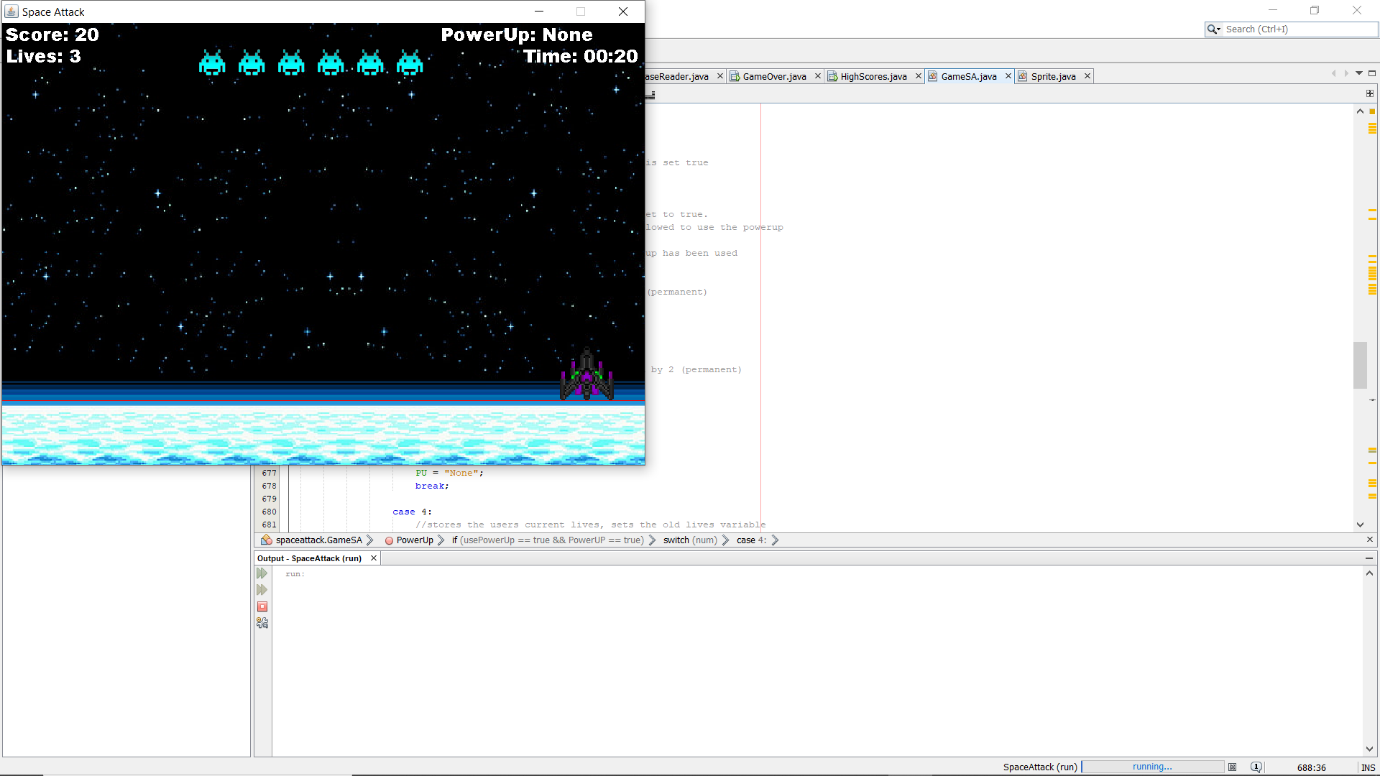
* ClearWave (Clears all current aliens on screen (Expendable))
* Speed (Increases the speed of the users ship by 2 (Permanent))
* BulletVel (Increases the speed of the users’ bullet by 2 (Permanent))
* Immunity (Gives the user immunity for several seconds (Expendable))
* AddBullet (Allows the user to shoot from all ship cannons for a limited time (Expendable))
* Health (Gives the user 2 added lives (Expendable, permanent))

PowerUps are used by the user to get stronger as the game increases by use of permanent powerups, used to clear aliens quicker (Expendable) or to survive longer (time based).

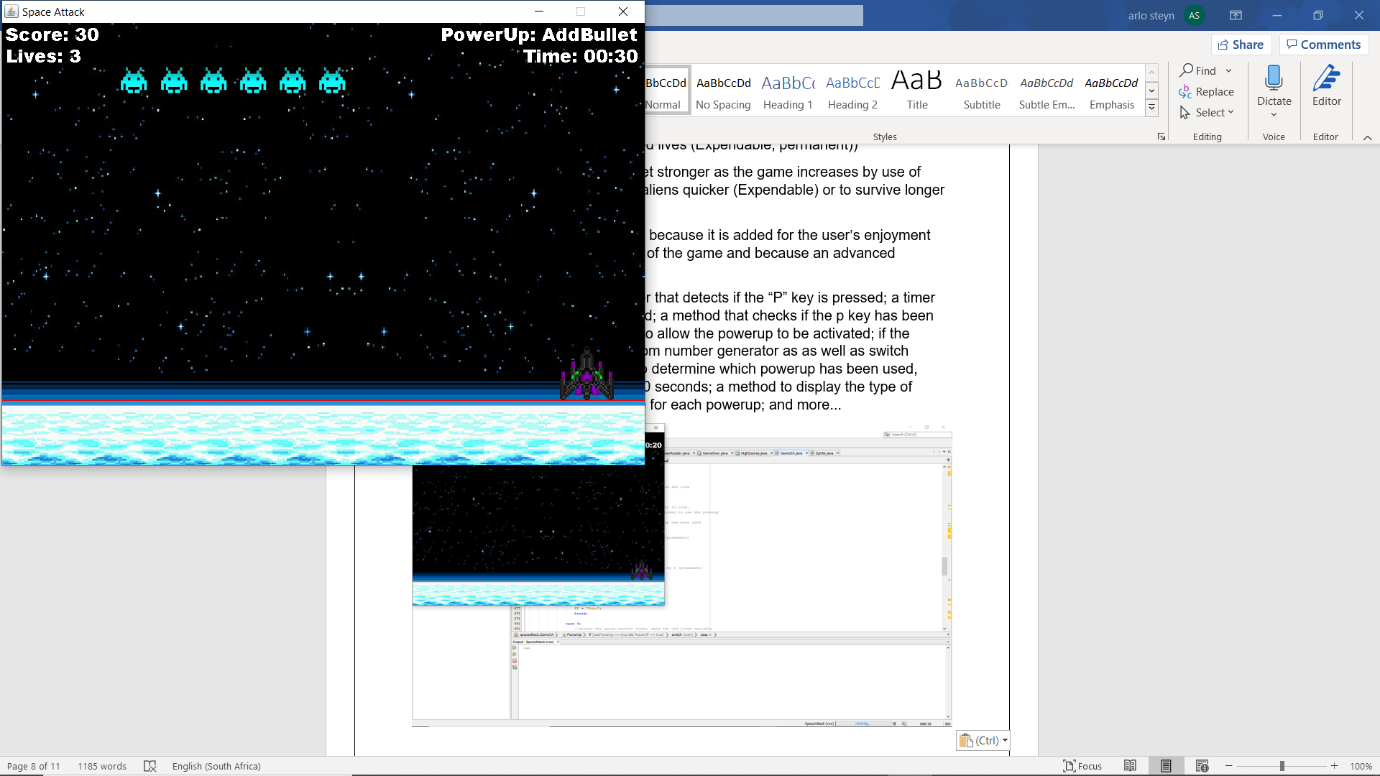
PowerUps is an advanced technique because it is added for the user’s enjoyment and not needed for the core function of the game and because an advanced algorithm was created for it.

The Algorithm includes: a key listener that detects if the “P” key is pressed; a timer that detects if 30 seconds has passed; a method that checks if the p key has been pressed if 30 seconds have passed to allow the powerup to be activated; if the powerup has been activated, a random number generator as as well as switch statements has been implemented to determine which powerup has been used, when used the timer waits another 30 seconds; a method to display the type of powerup to the user; 6 methods, one for each powerup; and more...

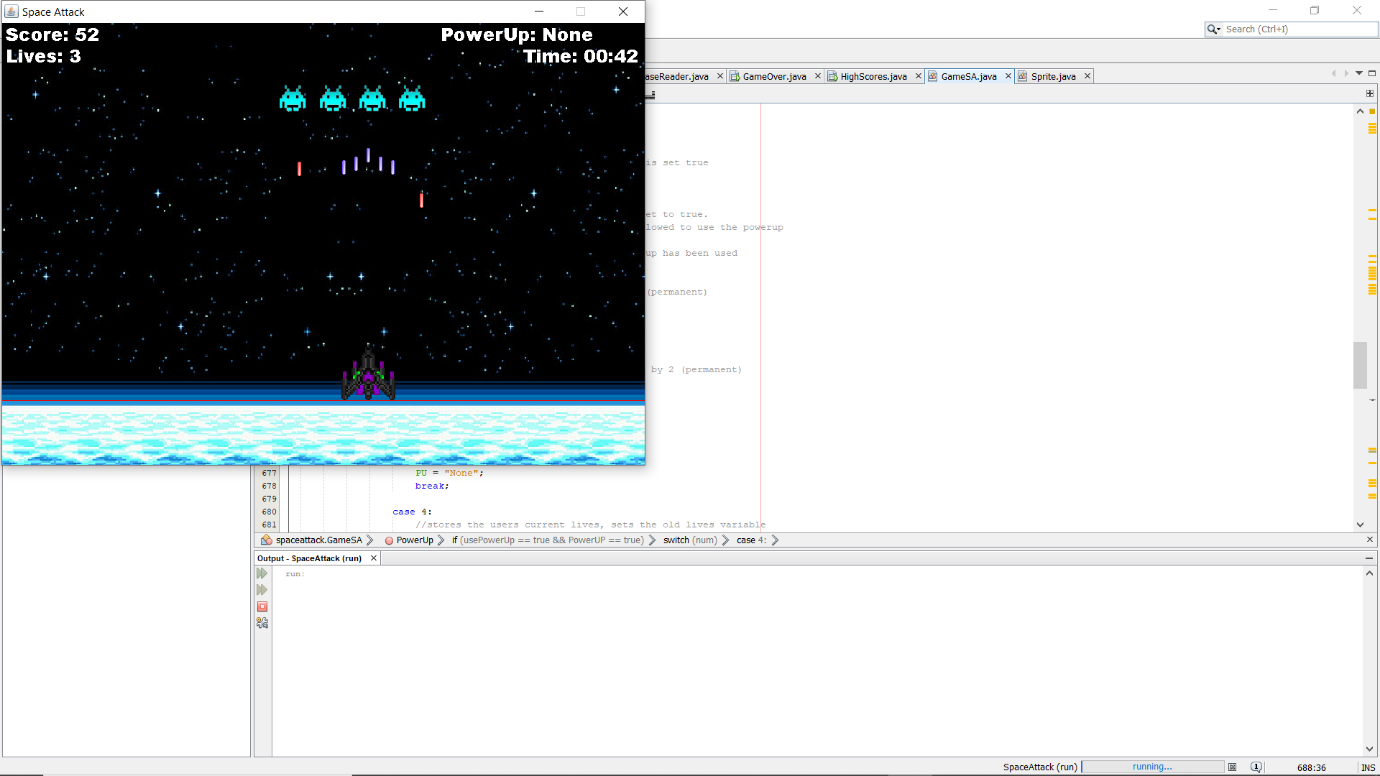
Example of how the advanced technique (power up)works



The timer is not equal to 30 seconds yet, therefore no powerup has been picked yet and it cannot be used



The timer is equal to 30 seconds, a powerup has been randomly chosen, “AddBullet” has been chosen by the AI but the user has not yet initialised the powerup, thus the “AddBullet” text remains



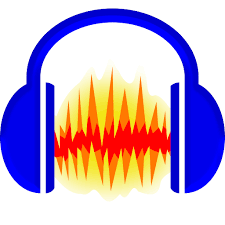
The user has pressed P and used the powerup, the text has changed back to “None” because there are no available powerups at the moment.

As you can see, the powerup “AddBullet” has worked as the user shoots 5 bullets at a time (Lasts several seconds)

**Music**

Music is added for the enjoyment of the user and to add on the retro gaming feel of Space Attack. This is an advanced technique because it is not needed for the functionality of the program, it only enhances it.

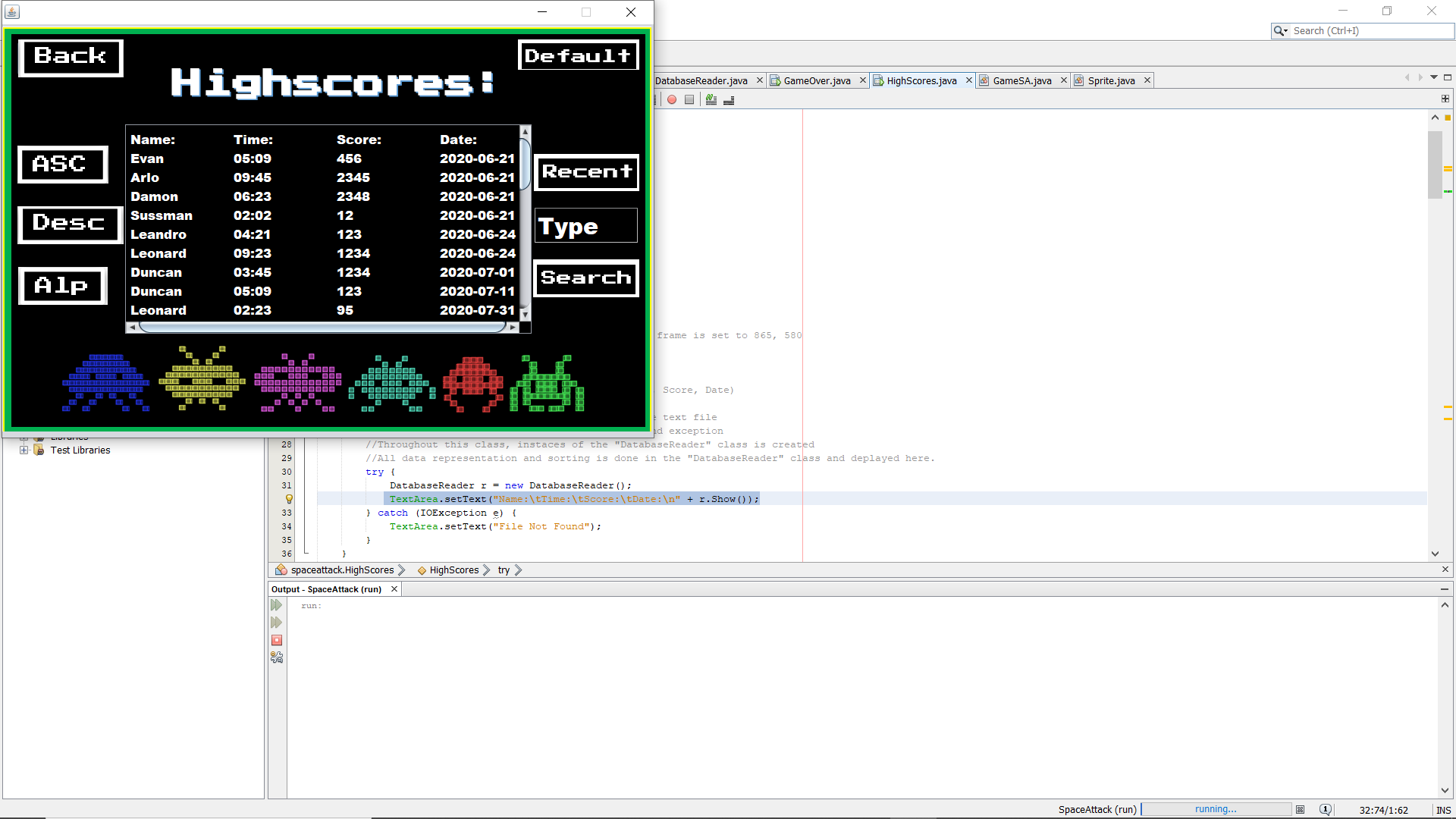
The following music was added:

* Danger Zone – Kenny Loggins.
* Shoot to Thrill – ACDC.
* We didn’t start the fire – Billy Joel.
* Perturbator – Future Club.
* Money for Nothing – Dire Straits.

These songs were added together by the use of “Audacity” into one large music file (“MusicSA”, 26 Mins long) as stated in Phase 2. Each song was chosen due to it fitting the theme, downloaded and added together and the implemented into the program. You are also able to toggle the music on and off in the Options screen.

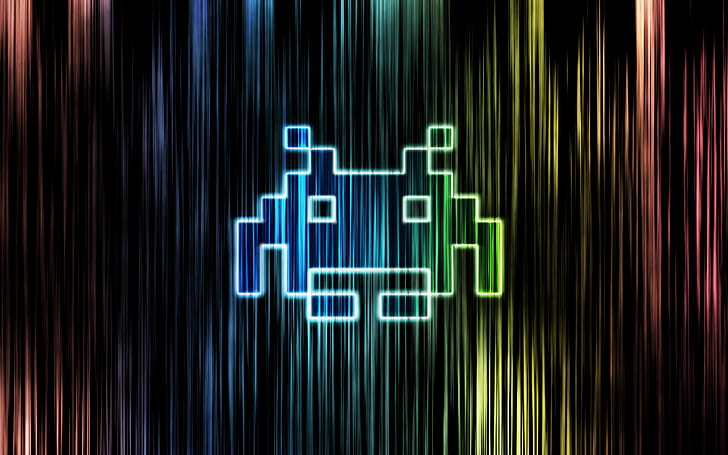
**Database/Highscores**

The Space Attack program allows the user to input their name before playing and display all their data in the “GameOver” screen once the game has ended. The database described in phase 2 shows that data can be sorted in many ways depending on what the user requires.



As you can see in the highscores screen, there are many ways to find and sort data. This is an advanced technique because; it was only required to have a database with basic viewing functionality, I have implemented several sorting and searching techniques that were not required. These databases methods add to the users experience of the program as they can sort and search for data depending on what they want to find.

The following sorting techniques were added:

* “ASC” – data in ascending order according to score.
* “Desc” – data in descending order according to score.
* “Alp” – data in alphabetical order according to name.
* “Default” – data displayed just as it is in the database.
* “Recent” – data in descending order according to date.
* “Search” – data displayed according to the name that it compares to.

**Testing documentation**

**Test plan and results**

The Space Attack programs database only requires the users name to be inputted each time that the user plays the game. All other data is generated by the AI based on how the user does in the game (score, time survived) and date played. Therefore, only one data validation check occurs, this is when the user enters his/her name. The user may only enter a name between 1 and 15 characters long. If this is not done, the name is not accepted and “Invalid!” is returned to the user. The users name may also not be equal to “Invalid!” as that is the message returned.

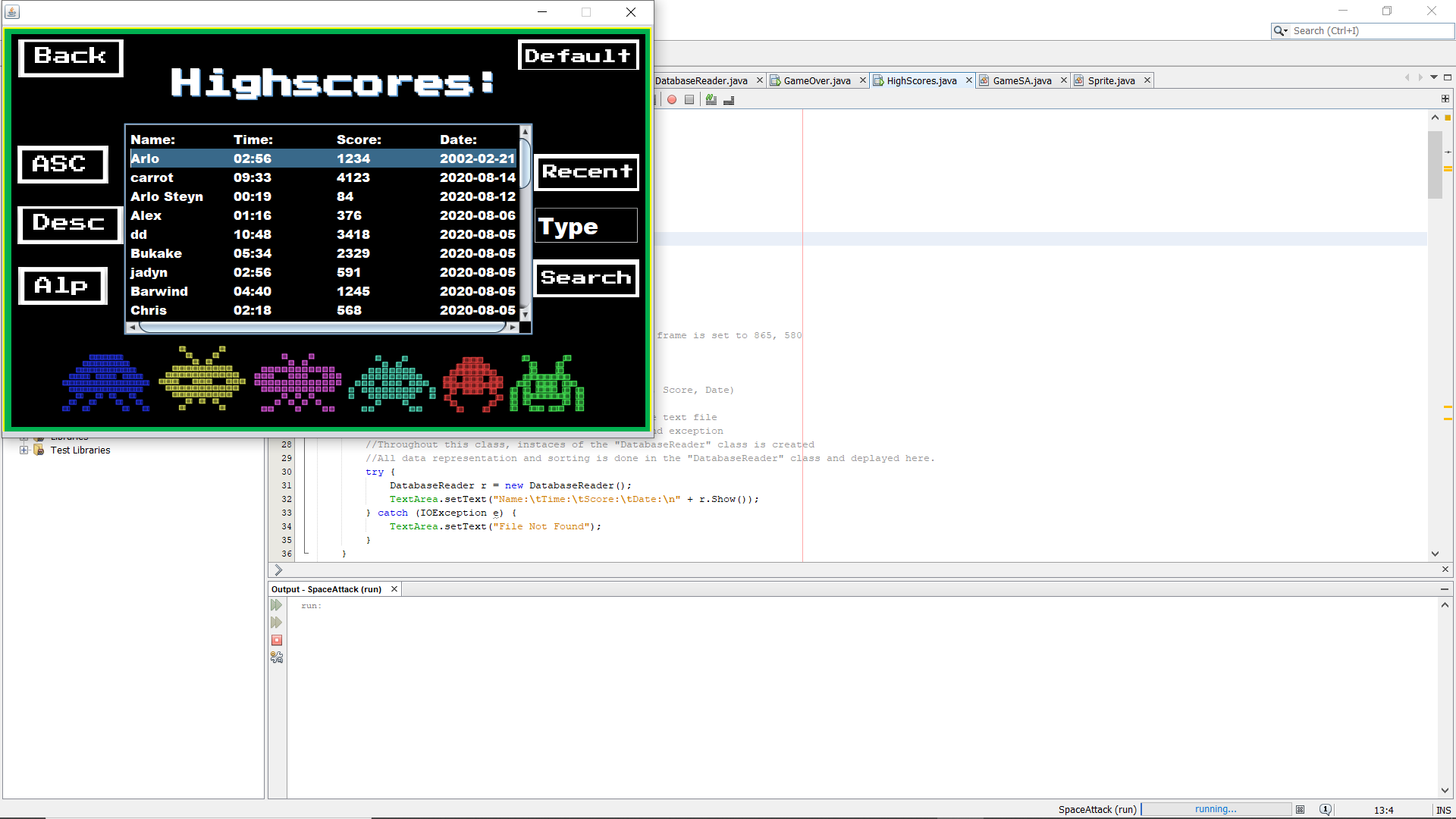
The variable that stores the time survived is in 00:00 format (minuets + seconds), therefore if the user survives for more than an hour, which he/she wont, the timer will reset and not record the hour.

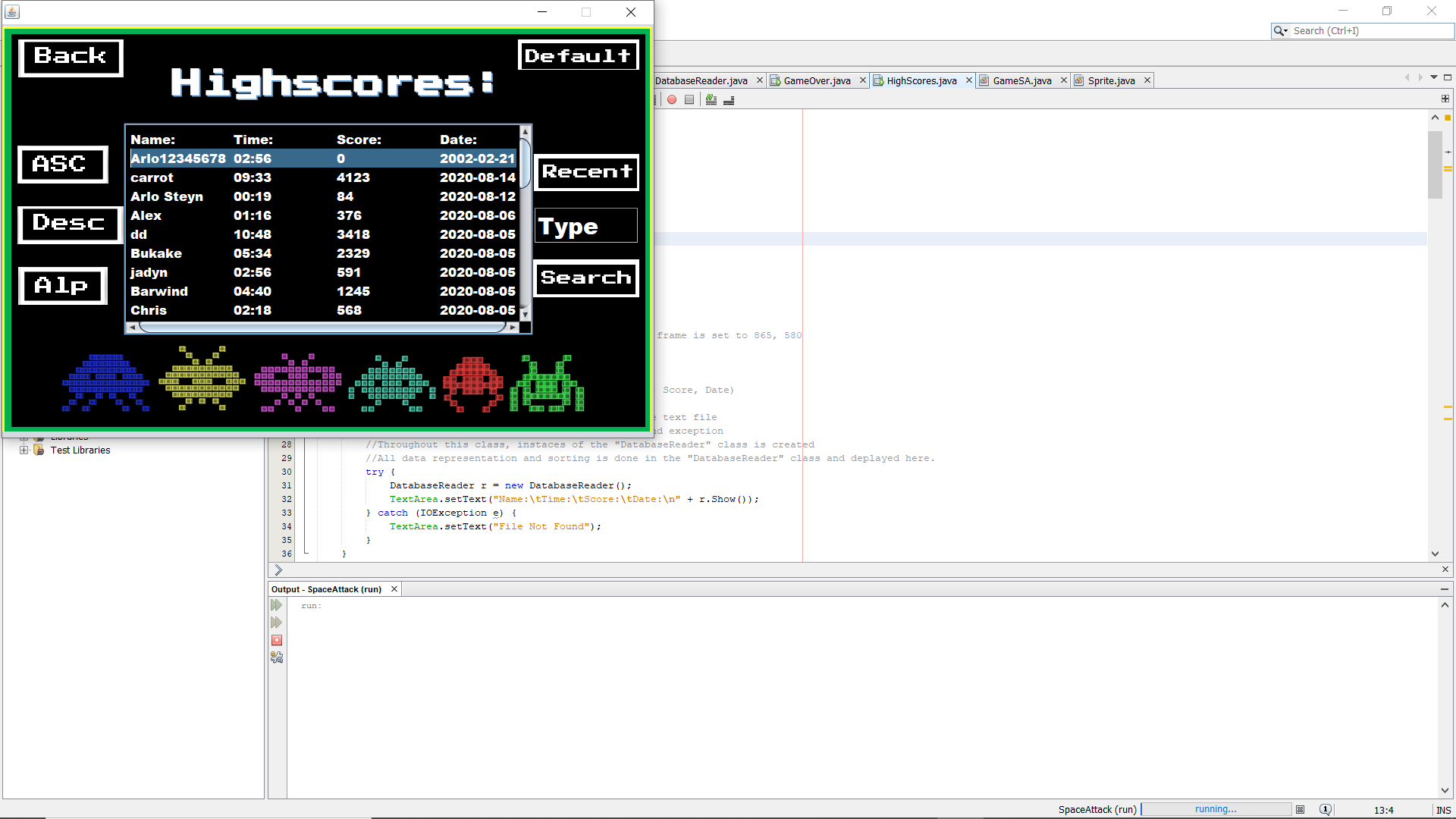
The date variable is stored depending on what day it is when the user plays, the date will therefore never be incorrect.

The score variable is dependant on how well the user does in the game because score is earned by killing aliens and every time the timer is incremented.

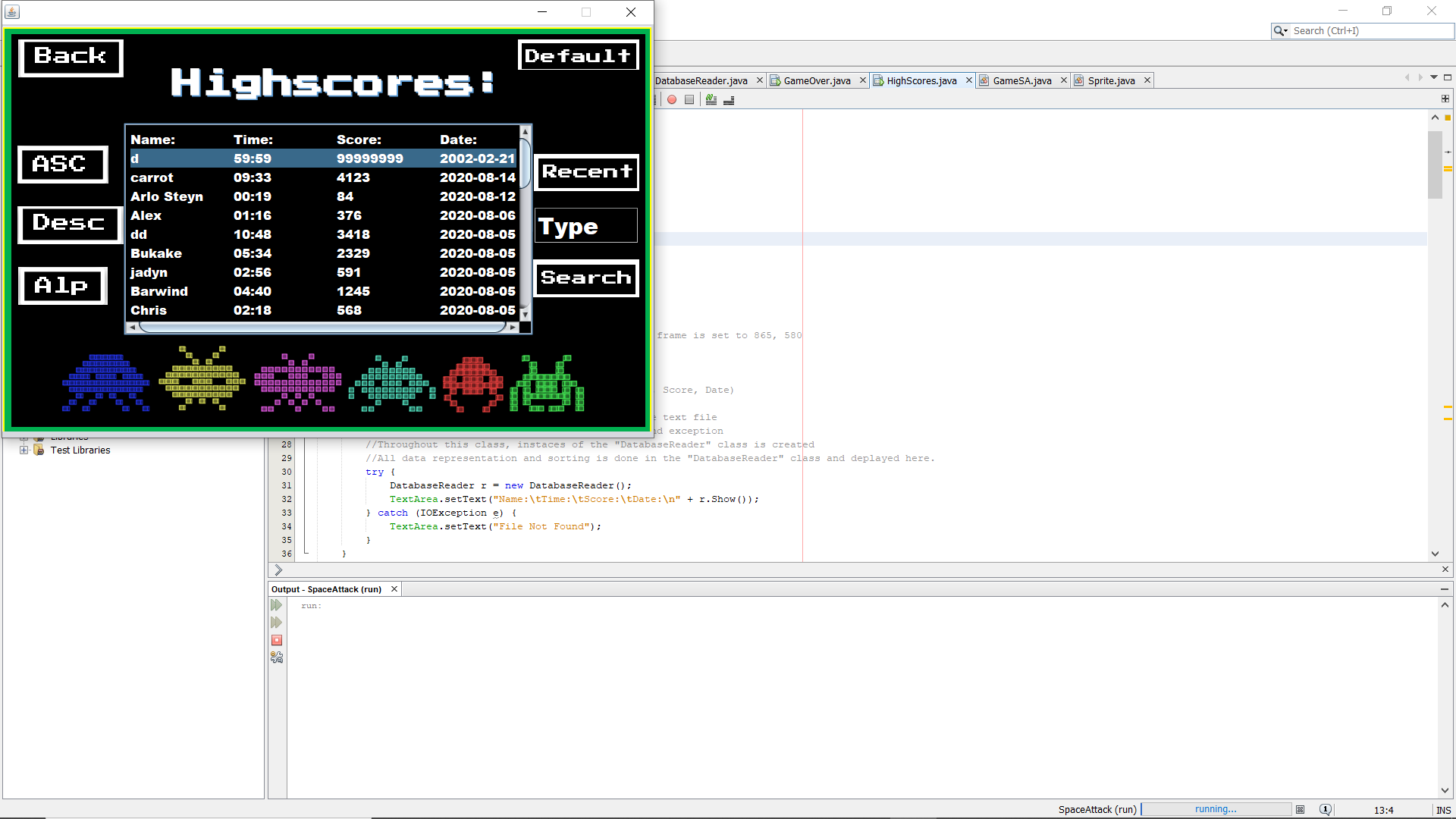
**Tested data and results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data input: | Data Type: | Normal: | Severe: | Extreme: |
| Name | String | Arlo | Arlo12345678 | d |
| Time | String | 02:56 |  | 59:59 |
| Score | int | 1234 | 0 | 99999999 |
| Date | String | 2002-02-21 |  |  |

Normal Test results:

Severe Test results:

The Name is 14 characters, 1 away from too many, however the program is still able to display it without breaking format.

Extreme Test results:

Extreme data still passes in program, as stated above if the player goes over an hour, the timer is reset to 00:00 and carries on.

