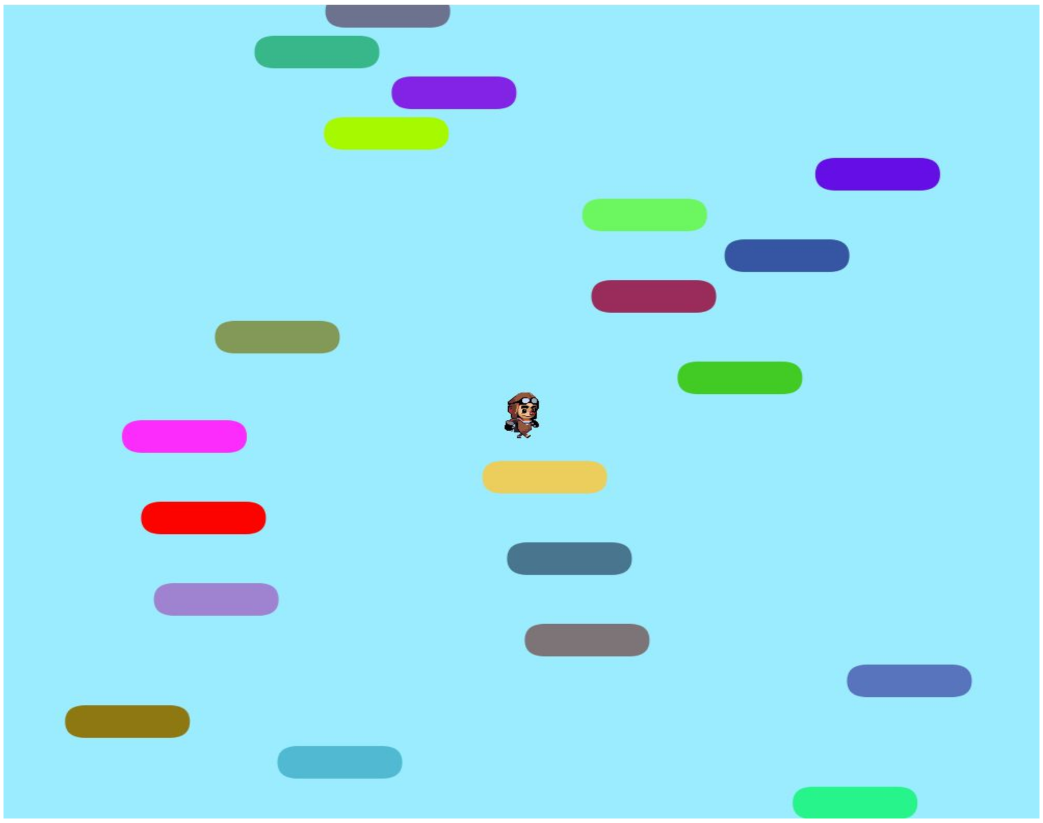
Project - Platform Hopper

This project will pull together a lot of the things you have learned up to now. You might want to have your old sheets and programs on hand to remind you of some things (especially material to do with *ArrayLists* and *Classes*)!

This is ***your*** game - after you have finished the basic version feel free to make any changes you want!

What we are aiming for:

We want to make a game very similar to the popular game app “Doodle Jump”. The aim is to guide an animated player sprite up a never-ending series of platforms without falling, with the left side of the playing field being connected with the right side.



What we are given:

- Player Class - a class that contains all of the variables (*xPos*, *yPos*, *xVel*, *yVel*, *width*, *height*, etc.) and methods (*move()*, *display()*, *animate()*, etc.) associated with the player in the game.

- Platform Class - a class that contains all of the variables (*xPos*, *yPos*, *yVel*, *width*, *height*, etc.) and methods (*move()*, *display()*) associated with the platforms in the game.

- PlatformHopper - the main window of the project that will call the player and platform methods (and any other class methods that we create later) in order to run the game.

  Note: Some of the functions have been purposefully left incomplete, with **pseudocode**. You will need to translate this pseudocode into working code blocks to get the project to run properly.

- Images to use for the player sprite and enemy sprite.

You must download the "Platform Hopper - Sample Project Code" zip folder from the Academy Of Code resources webpage - http://www.theacademyofcode.com/resources.

Steps to make the basic game

To make a simple version of the game we need to:

1. **Translate the pseudocode** in the functions located in the main window of the project.
2. Call the relevant **functions** and **class methods** in ***void draw()*** to run the game.

Once you have that - congratulations on completing the basic version of the game! There's plenty more you can add now...

Extras

- Add a score and show it on the screen.
  - Maybe the player gets 1 point per platform that goes off the screen?
  - Perhaps there's a bonus every 10 or 20 platforms? (Your decision!)

- Improve the game over screen.
  - New colours?
  - New text?
  - Use an image instead of a background?
  - Press a button to restart the game?

- Add platforms that move left and right.
  - You can have every 15th platform moving or you can set the chance of a platform moving to 10% (this can be done by generating a random number between 1 and 100 and set the platform to moving if the random number is between 1 and 10).

- Add in levels:
  - This could involve increasing the level when the player's score reaches multiples of 100, for example.
  - Maybe the background colour changes and the difficulty increases (less platforms on the screen and/or the platforms move faster).

- Add "weak" platforms that "break" when the player touches it.
  - Use the same algorithm as the task above to choose the "weak" platforms.
  - You can have it so that as soon as the player touches the "weak" platform, the player does not bounce and the platform is deleted from the *ArrayList*.
  - Tell the player that it is a "weak" platform by using some text or have a thick outline, for example.

- Add "power-up" platforms that will launch the player up the screen.
  - This can be accomplished by increasing the velocity of the player in the y-direction (*yVel*) when the player bounces on a "power-up" platform.
  - Tell the player that it is a "power-up" platform by placing an image of a spring on top of the platform, for example.

- Add in enemies on top of some of the platforms that the player has to avoid.
  - Use the same algorithm as the left to right platforms and the "weak" platforms to choose when an enemy will spawn on a platform.
  - If the player collides with an enemy, the game is over.
  - If the player collides with an enemy in mid-air while the player's velocity in the y-direction (*yVel*) is positive (the player jumps on top of the enemy), then the enemy will be killed (removed from enemy *ArrayList*).

- Add in the ability for the player to fire projectiles.
    - The projectile fires when the player presses a button (the space bar is probably the most intuitive).
    - The projectile spawns at the centre-point of the player.
    - If the projectile collides with an enemy, then the enemy is killed (removed from enemy *ArrayList*).
    - There should be a reload time so the player can only shoot one projectile every second/half a second for example.