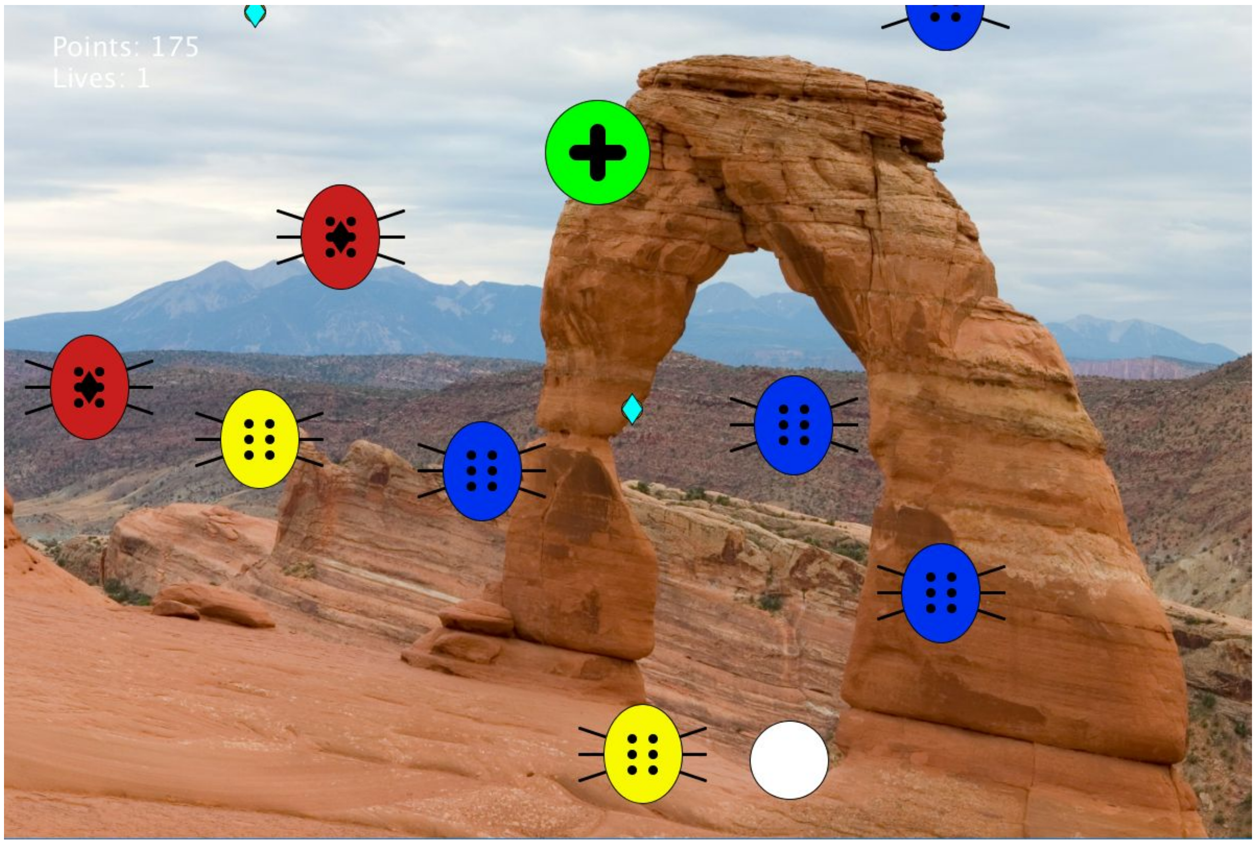
The Bug Catcher

Back at the start of your time with the Academy of Code you made a game where you had to dodge a falling ball. You've learned a lot since then!

In this lesson you are making a game which shares a lot of the basic elements of that game, but which is vastly more sophisticated. You will make use of **classes**, which will allow us to include large numbers of different object types on-screen at once. The object of the game is to catch the good bugs and avoid the bad bugs. There are also some extra life tokens which fall from time to time.

The below is a screenshot of the game in action:



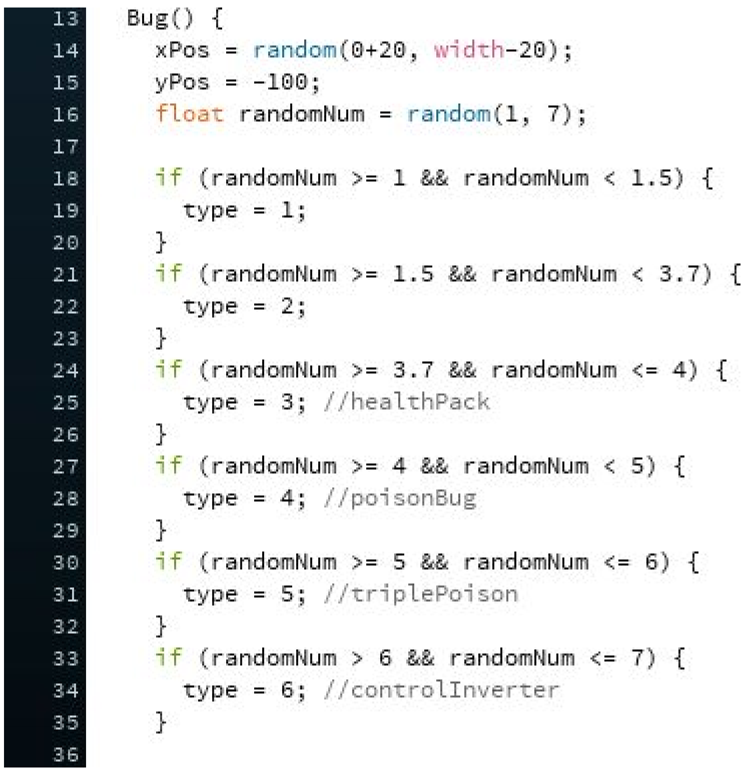
The white circle is the player (you should find something more visually interesting for the player to control). The other objects are all described on the final page of this handout (feel free to make design improvements!).

To get started, here are a few of the first things you'll need to do:

- Create a *Bug* class
- Create a *Player* class
- Add keyboard controls (ideally within a function inside the *Player* class - maybe "void move()"?
- Create a single object of type *Bug* and make it fall down the screen. Detect collisions with the player
- Add a game over screen if the players lives drops to 0 or below

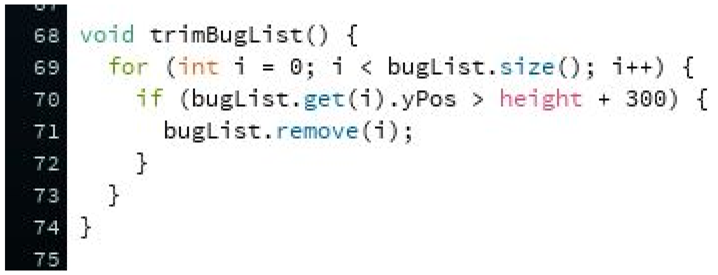
Once you're happy with all that, here is some more detail on what you will need to implement:

- A cool background (you can definitely do better than the one in the screenshot!)
- *Player* class should have:
    - Variable for position, size, points, lives, speed and *invertControls* (for use with the bamboo grub type)
    - Function "void move()" which has the usual "if (keyPressed)..." stuff in it
    - Function "void display()", which draws the player (you can optionally also put the text printing for points and lives in here
- *Bug* class should have:
    - Variables for position, speed, size, points bonus (can be positive or negative), health bonus (can be positive or negative), *invertsControls* (a boolean - set true for the bamboo grub type) and type (an int - each bug type has a number).
    - A constructor, which determines the bug type. The start of the constructor looks like this (can you make the random number bit more elegant?)



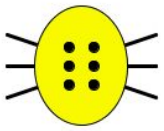
```
Bug() {
  xPos = random(0+20, width-20);
  yPos = -100;
  float randomNum = random(1, 7);

  if (randomNum >= 1 && randomNum < 1.5) {
    type = 1;
  }
  if (randomNum >= 1.5 && randomNum < 3.7) {
    type = 2;
  }
  if (randomNum >= 3.7 && randomNum <= 4) {
    type = 3; //healthPack
  }
  if (randomNum >= 4 && randomNum < 5) {
    type = 4; //poisonBug
  }
  if (randomNum >= 5 && randomNum <= 6) {
    type = 5; //triplePoison
  }
  if (randomNum > 6 && randomNum <= 7) {
    type = 6; //controlInverter
  }

```

    - This constuctor also needs to set some of the variables depending on which type the bug is (we did it with a string of ifs for type == 1, type == 2, etc
    - A "void move()" function, which adds speed to position
    - A "void display()" function, which draws the bug depending on its type
- In the main file you need an arrayList to store all of the bugs we're drawing. This is one of the *massive* differences between this and the project we did several terms ago! It allows us to have almost as many bugs as we want without writing extra code for each one. We declare the ArrayList with "`ArrayList<Bug> bugList;`", and then we instantiate it in void setup() with the line "`bugList = new ArrayList<Bug>();`".
    - Note that *Bug* is the name of the class - this needs to be exactly the same as the class name, including capital letters!
    - To add a new Bug to the ArrayList of bugs we use this line: "`bugList.add(new Bug());`"
        - We should do this every 60-120 frames (you can decide the exact number yourself). We did this with a timer variable, which we add 1 to every frame. The line "if (timer%90 == 0)" will be true every 1.5 seconds
        - We used a variable called *interval* instead of the 90 above - that let us make the bugs more frequent as play goes on. You can add this later, if you like
    - To loop through all the bugs use this line "`for (int i = 0; i < bugList.size(); i++) {...}`". You can then access a specific bug (for example, its healthBonus variable) with "`bugList.get(i).healthBonus`"
    - When you connect with a bug use the ".remove(i)" function to remove it from the list.
    - Be careful with the ones you don't catch! They'll stay on the list forever if you don't do something about it. Code like this will help:



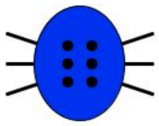
```
void trimBugList() {
  for (int i = 0; i < bugList.size(); i++) {
    if (bugList.get(i).yPos > height + 300) {
      bugList.remove(i);
    }
  }
}
```

    -



SAMURAI BEETLE

This yellow bug is our “basic” bug, which appears about 30-40% of the time. You get 5 points for catching one. Moderate speed.

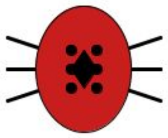


BLUE BEETLE

The blue beetle is rarer, appearing less than 10% of the time. It’s worth 15 points. Slower than the samurai beetle.

HEALTH PACK

Gives the player an extra life. Appears approx 5% of the time. Similar speed to the blue beetle



FIRE ANT

Avoid! Appears approx 15% of the time. Player loses a life and loses 25 points if they collide with a fire ant. Slightly faster than the samurai beetle.

ARMY ANT

Very small, very fast, and very dangerous. Appears approx 15% of the time. Takes away three lives, and costs 100 points.

BAMBOO GRUB

Very small and fast moving, being hit by one of these grubs causes the player to become disoriented, reversing their controls. (Either for a timed period, or until they catch another bamboo grub). Appears approx 15% of the time.

EXTRAS/IMPROVEMENTS/BUG FIXES

You are expected to add at least one more type of bug/power-up to the game (preferably many types!), in addition to any other improvement. There are also some issues you will run into in the game which you will want to address. Here are some ideas:

- Speed power up (player moves faster for some set period of time)
- Speed power down (makes player slower, or all bugs faster, for a set period of time)
- Random bug speed - we implemented a range of speeds for our bugs, so that different bugs of the same type can have different speeds
- Change size of player (either bigger or smaller, depending on what you catch)
- Save high scores (first within the game, then save to a file)
- Prevent the player moving over the side of the screen
- Add a new level, with a whole new set of bugs (or maybe just one or two extra in the new level