Australian National University

Structured Programming

COMP1110/6710

← Needs ANU Account!

pollev.com/fabianm
Register for Engagement

Image Courtesy NASA/JPL-Caltech.

Admin

- Reminder: Don’t e-mail us, we e-mail you.
  But if you must, use comp1110@anu.edu.au
- CWAC Submission Website Online – Try it at https://comp.anu.edu.au/courses/comp1110/cwac
- THIS WEEK ONLY: Deadline for Commit Selection extended to noon today
- Soon: Code Walk Registration, Code Walk Scheduling Preferences, more Precise Assignment Variable Tracking
- Reminder: Academic Integrity
- Reminder: Keep your repositories **PRIVATE**

- Code Walks: **Bring physical Student ID Card!**

Recap: Assignments

Three steps you must do:

- Push Code on GitLab (can use assignment variables/extensions)
- Register for Code Walk (Base Deadline Day, 18:00, no extensions)
- Attend Code Walk at Scheduled Time

Two optional steps:

- Provide Particular Commit You Want to Submit (otherwise, latest commit before base deadline)
- Provide Scheduling Preferences for Code Walk (otherwise, some time during your registered tutorial time)

Practice

Fork and clone the comp1110-2025s1-workshops project.
Create a folder “ws3b”, and work in “FallingMarbles.java” in there.
Commit and push when you are done.

Following the design recipe, design a world program that runs on a 800x800 pixels WHITE background square and behaves as follows. Each time the user left clicks with the mouse on the screen, the program draws a new marble (e.g., a circle of say, radius 20, or otherwise an image of your preference) with center located at the position of the click and randomly chosen colour among the following 5 possibilities: RED, GREEN, BLUE, MAGENTA, or BLACK. The marbles must fall down at a constant speed of 5 pixels/step and disappear from the window once the reach the bottom of the background image.

In a first version of the program it is ok if memory consumption grows arbitrarily as we left click with the mouse. However, in a second stage, you may also want to develop an improved version which reduces memory consumption by removing those marbles which disappear from the screen.

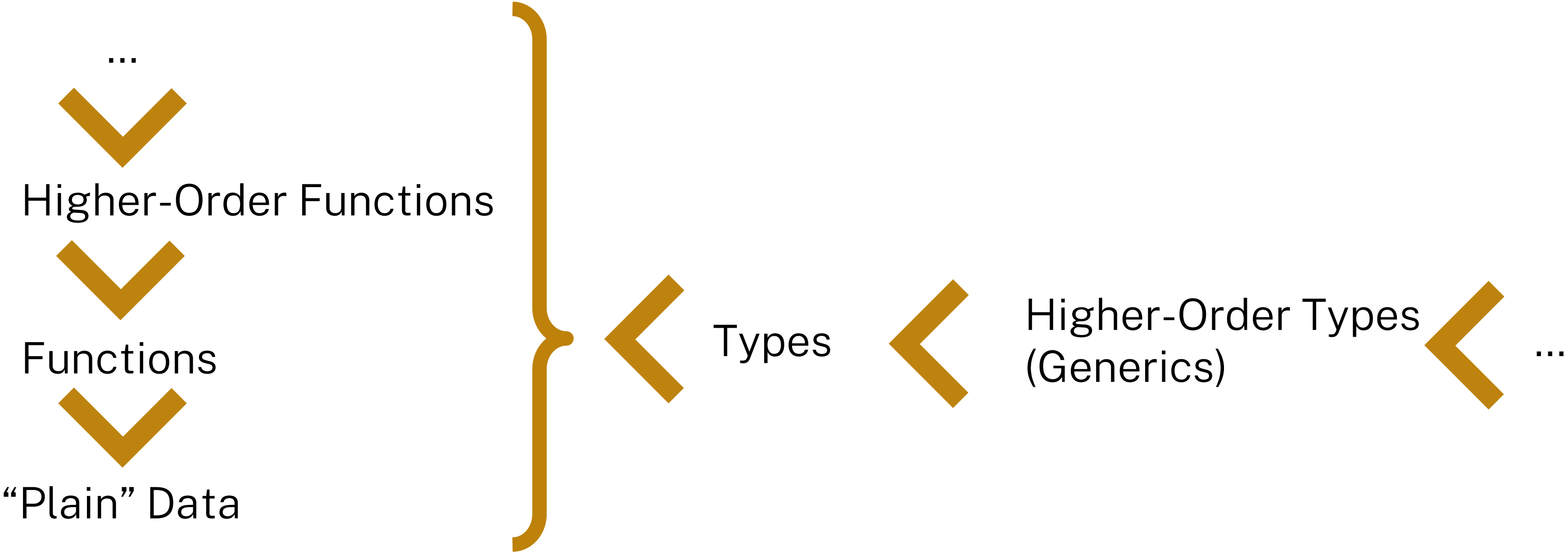
Higher-Order Functions

Abstraction, Part 4

Australian
National
University

“Higher-Order”?

Abstractions come in Hierarchies:



...

Higher-Order Functions

Functions

“Plain” Data

Types

Higher-Order Types (Generics)

...

Recap: The ConsList<T> Template

```
// { ...
//   return ... switch(list) {
//    case Nil<T>() ->... ;
//    case Cons<T>(var element, var rest)->
//        ... element ... [recursiveCall](... rest ... ) ... ;
//   } ...;
// }
```

Key Motto:

The shape of the data determines the shape of the code!

Fold (Right)

```
<S,T> T Fold(BiFunction<S, T, T> agg, T base, ConsList<S> list) {
  return switch(list) {
   case Nil<T>() -> base;
   case Cons<T>(var element, var rest)->
       agg(element, Fold(agg, base, rest));
  };
}
```

Fold (Left)

```
<S,T> T FoldLeft(BiFunction<S, T, T> agg, T base, ConsList<S> list) {
  return switch(list) {
   case Nil<T>() -> base;
   case Cons<T>(var element, var rest)->
       Fold(agg, agg(element, base), rest);
  };
}
```

Map

```
<S,T> ConsList<T> Map(Function<S, T> mapper, ConsList<S> list) {
  return switch(list) {
   case Nil<S>() -> new Nil<T>();
   case Cons<S>(var element, var rest)->
        new Cons<S>(mapper(element), Map(mapper, rest));
  };
}
```

```
OR: Fold((s,t) -> new Cons<T>(mapper(s), t), new Nil<T>(), list);
```

Filter

```
<T> ConsList<T> Map(Predicate<T> pred, ConsList<T> list) {
  return switch(list) {
   case Nil<T>() -> new Nil<T>();
   case Cons<T>(var element, var rest)->
       pred(element)?new Cons<T>(element, Filter(pred, rest))
                    : Filter(pred, rest);
  };
}
```

```
OR: Fold((s,t) -> pred(s)?new Cons<T>(s,t) : t, new Nil<T>(), list);
```

Exercises

Fork and clone the comp1110-2025s1-workshops project.
Create a folder “ws3b”, and work in “Lists.java” in there.
Commit and push when you are done.

1. Go to the Standard Library Documentation on the Course Website, and find the List functions.
2. Do the following exercises with and without the higher-order functions.
3. Try implementing the following functions yourself (use lower-case names):
   `IsEmpty, First, Rest, Length, Nth, Append, BuildList`
4. Implement the function `snoc`, which appends an element at the end of a given list
5. Implement the function `reverse`, which reverses the order of the elements of a list.

Practice

Fork and clone the comp1110-2025s1-workshops project.
Create a folder “ws3b”, and work in “FallingMarblesHO.java” in there.
Commit and push when you are done.

Like the FallingMarbles exercise above, but this time with higher-order functions.

You can add additional functionality:

- Pressing keys “R”, “G”, “B”, “M”, or, “L” removes all RED, GREEN, BLUE, MAGENTA, or BLACK marbles, respectively
- Pressing arrow keys moves all marbles 5 pixels in the corresponding direction
- Right-clicking a marble changes it’s colour to a different random colour