GPXWizard (GWiz)

Katherine Duff (kpd2128)
Ashley Kim (atk2141)
Elisa Luo (eyl2130)
Rebecca Yao (rby2107)

February 3, 2021

Introduction

"Gee-whiz"
adjective: gee-whiz
characterized by or causing naive astonishment or wonder, in particular at new technology.'

The GPXWizard, or GWiz, is a programming language that allows for the modification and analyzation of GPX files. GPX files are generated by using a watch, phone, or other device to track a run, walk, swim, hike, or bike ride.

A GPX file is a GPS file in GPS exchange format and is saved in XML file format. It stores information such as waypoints, tracks, and routes. Waypoints (<wpt>, </wpt>) are the simplest form of data in a GPX file, as they only store latitude and longitude. Tracks (<trkpt>, </trkpt>) consist of a series of waypoints with time stamps. Between every track point is a straight line, so the more track points recorded, the more accurate and smoother the track will be. Routes (<rtept>, </rtept>) are a series of waypoints, where each waypoint is a significant landmark.

All GPX data interpretation and adjustments can be completed in the Activity class built into GWiz. Basic syntax will most closely resemble Java. GWiz is object oriented, allowing a user to compare 2 Activity objects where each Activity is 1 GPX file. Activity is similar to the LinkedList ADT, as this is where waypoints, trackpoints, or route points will be stored sequentially.

Purpose

Have you ever gotten so infuriated during a run you threw your phone across the road, only to later discover your digitally tracked running path had a bump in it because you forgot to pause activity when you tossed your phone?

Probably not since most of us are computer scientists and are allergic to the outdoors, but if you happened to experience this, GWiz can fix it.

GPS isn't perfect. GWiz allows for a user to retroactively edit points in their route. Or if a user accidentally stopped an activity and had to start a new one, GWiz can help them merge the two. A user can also easily compare multiple activities they completed to track their progress. Many fitness tracking apps allow for limited modifications to an activity, but by allowing a user to toy directly with the source GPX file, GWiz has a plethora of ways a user can edit and analyze their run/bike/walk.

Syntax

Data Types

Primitive Data Types

A primitive data type specifies the size and type of variable values, and it has no additional methods.

Type	Size	Description	Examples
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647	`99999999999999`, `17, 420`
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits	`0.66666666667`, `3.14159265`
bool	1 bit	Stores true or false values	`True, False`
char	2 bytes	Stores a single ASCII character	`'F', '@', ';'`

Dynamically Allocated Types

Type	Description	Example
String	An array of chars	`String s = "hello";`
Array	Sequential block of variables of the same type	`arr[3] = {3,6,9}`

Complex Types

Built-in types that are represented as objects.

Type	Description	Constructor Parameters	Methods/attributes
GPXFile	An abstract representation of file and directory pathnames. The original GPX file is read-only.	**filepath** (String) - the absolute file path to the .gpx file	Attributes: *filepath*
GPXScanner	A simple text scanner that can parse a GPX file using regular expressions.	none	**read_gpx** (GPXFile file) - parses the .gpx file specified by the file parameter
Coordinate	Representation of a single coordinate of longitude and latitude on Earth.	**longitude** (double) **latitude** (double)	Attributes: *longitude, latitude* Accessor functions only
DateTime	Representation of a single naive date and time.	**UTCDateTime** (String) OR **year** (int), **month** (int) ... etc.	Attributes: *year, month, day, hour, minute, second* Accessor functions only
TrackPoint	Representation of a single GPS waypoint (Coordinate & DateTime)	**coor** (Coordinate) **dt** (DateTime)	Attributes: *nextPoint, prevPoint, dist2next, time2next...etc* **calc_time2next()** - returns the timeDelta to the next TrackPoint **calc_dist2next()** - returns the distance to the next TrackPoint
Activity	A linked list of TrackPoints	**head** (TrackPoint) - the first TrackPoint in the linked list	Attributes: *name, total distance, total time, head, tail* **calc_tot_dist()**, **calc_tot_time()**

Separators

- parentheses ()
- curly brackets {}
- square brackets [ ]
- semicolon ;
- comma ,
- period .
- double quotes " "
- single quotes ' '

Control Flow

Keyword	Usage	Example

if	Handles conditional statements	if(x==5){
		print(”x is 5”);
		}

for	Handles loop operations	for (int i=0; i<5; i+=){
		print(i);
		}
		//prints 0 1 2 3 4

while	Handles loop operations	int x = 0;
		while (x<10){
		print(x);
		x+=2;
		}
		//prints 0 2 4 6 8

continue	When a continue statement is encountered inside a loop, control jumps to the beginning of the loop for the next iteration	for (int i=1; i<10; i++){
		if (i==5){
		continue;
		}
		print(i);
		}
		//prints 1 2 3 4 6 7 8 9

break	Terminates a loop based on a condition, continues at next statement	for (int i=1; i<10; i++){
		if (i==5){
		break;
		}
		print(i);
		} //prints 1 2 3 4

Operators

Operator	Usage	Example
+, -, *, /, %	Addition, Subtraction, Multiplication, Division, Mod	1 + 2 = 3 3 - 9 = -6 7 * 8 = 56 5 / 1 = 5 30 % 4 = 2
=	Assignment	x = 2;

+=,
-=,
=,
/=
	Addition assignment,
	Subtraction assignment,
	multiplication assignment,
	Division assignment
		x += 3; // x = x + 3
		x -= 7; // x = x - 7
		x *= 5; // x = x * 5
		x /= 3; // x = x / 3

==,
!=
	Equality comparator for primitive types,
	works for all built in types
		8 == 8; //returns True
		8 != 9; //returns True

&&, ||, !	Logical AND, OR, NOT	x <5 && x <10 x <5 ||x<3 !(x<4)

<,
>,
<=,
>=
	Less than,
	Greater than,
	Less than or equal to,
	Greater than or equal to
		for x=3...
		x>4; //returns False
		x<=3; //returns True
		x>=5; //returns False

[]	Index	array[0] = 100; // initializes first element of array to 100

Miscellaneous

The language has a similar comment structure to Java

- // This is a single line comment

- /*
  This is a multi
  line
  comment
  */

Keywords

- null: a complex type with no value
- void: designates a function without a value to be returned
- Run, Walk, Swim, Bike: designates Activity type

- main: designates the main function
- for: looping structure
- while: looping structure
- if, else if, else: conditional control flow
- true, false: for bool type, designates true and false values
- throw: designates when an Error object should be instantiated and returned
- return: end of function, followed immediately by value to be returned if applicable

User Defined Functions

Example 1

```
bool greaterThanTwo (int a) {
   return (a > 2);
}
```

Example 2

```
int gcd (int a, int b) {
   int remainder = 0;
   while (a % b > 0) {
      remainder = a % b;
      a = b;
      b = remainder;
   }
   return b;
}
```

Sample Program

Compare Distance

This program takes two .gpx files and prints out the longer of the two distances travelled.

```
void main() {
// Create two GPXFile objects with .gpx files taken from a run
GPXFile jan30 = GPXFile(user2_01_30_2021.gpx);
```

```
GPXFile jan31 = GPXFile(user2_01_31_2021.gpx);

// Create GPXScanner object to parse the two GPXFiles
GPXScanner reader = GPXScanner();

// Parse the GPXFile using the GPXScanners method read_gpx()
Activity satRun = reader.read_gpx(jan30);
Activity sunRun = reader.read_gpx(jan31);

// Use the Activity objects distance method to get distances for each
double satDist = satRun.calc_tot_dist();
double sunDist = sunRun.calc_tot_dist();

    // Compare total distances and print distance of the longer activity
    if (satDist > sunDist) {
       print(satDist);
    } else {
       print(sunDist);
    }
}
```

Change Starting Point

This program parses a .gpx file, then changes the starting latitude, longitude, date, time and prints out the new total distance.

```
void main() {
   // Create a GPXFile object
   GPXFile file1 = GPXFile(user3_01_21_2021.gpx);

   // Create a GPXScanner object to parse the GPXFile
   GPXScanner reader = GPXScanner();

   // Parse the GPXFile using GPXScanners method read_gpx();
   Activity goldenRun = reader.read_gpx(file1);

   // Create new TrackPoint object with a coordinate and DateTime
   Coordinate c = Coordinate(0, 0);
   DateTime dt = DateTime(2021-01-21T23:30:42Z);
   TrackPoint tp = TrackPoint(c, dt);

   // Modify the head of the Activity to be the above TrackPoint
   goldenRun.head.data = tp;

   print(goldenRun.calc_tot_dist());
}
```

Conclusion

Inspiration for this project was taken from a few of the group member's religious use of the popular fitness app Strava. We also wanted to pay homage to the introductory programming language at Columbia University, Java, so created Java + Strava = GPXWizard. Don't ask how that string addition (not concatenation) worked.