Homework 1

Arik Rinberg 317156669

Barak Zan 305634487

Question 1

Subsection b.1

The implementation will have to be changed to either add gs or $gs.reverse()$, based on if $gs.p1 == this.end$ or $gs.p2 == this.end$ respectively.

Subsection b.2

This specification is stronger, as the $requires$ is more lenient, meaning we allow more inputs to the function.

Subsection d

We'll use the definition from the lectures: B is a subtype of A iff an object of type B can pretend to be an object of type A in any context.

$Route$ is **not** true subtype of $GeoFeature$, as $GeoFeature$ has a $name$, and $Route$ does not.

$GeoFeature$ is **not** a true subtype of $Route$, as $Route$ allows adding a $GeoSegment$ with name X and then a $GeoSegment$ with name Y, and $GeoFeature$ does not.

Subsection e

A class that can inherit from $RouteFormatter$ is $BicycleRouteFormatter$ that formats the directions into sentences that looks like:

"Turn slight right onto Hankin Road and cycle for 27 minutes."

It is obvious why $BicycleRouteFormatter$ can inherit from $RouteFormatter$.

A class that cannot inherit from $RouteFormatter$ is $FlyingRouteFormatter$ that given a list of geographical features, creates a new segment from $Route.start()$ to $Route.end()$, and flies directly there. The class will format the $Route$ into a single sentence that looks like:

"Turn to heading 137 and fly for 100 minutes."

$FlyingRouteFormatter$ cannot inherit from $RouteFormatter$, as $RouteFormatter$ prints a line per $GeoFeature$, and $FlyingRouteFormatter$ prints a single line for the entire route.

Question 2

Subsection b

If we have an infinite amount of segments in our route, holding it all in memory will require an infinite amount of memory.

In the future, we may choose to upload the entire route to the cloud, and just hold a subsection of the route of the first few segments that the user needs to pass, and as the user passes segments they will get flushed and new ones will be pulled from the cloud keeping memory space bounded.

Subsection c

The two parameters are the $Frame$ and the $Parent$.

The $Frame$ holds multiple $dialog$s, and specifies which one is currently active. Thus we notice that after clicked "Add GeoSegment" and the new dialog window opens, we cannot click the button again, or even make changes the entire dialog window. If we want two or more concurrently interactive dialog windows, we must open more $Frames$.

The $Parent$ is used in order to be able to pass data back from one dialog window to another. As if we had multiple windows of $RouteFormatterGUI$ open at the same time, without the $Parent$ we wouldn't be able to know to which $route$ to add the $segment$ to.

Subsection d

We would first need to add a function to $Route$ which allows the removal of a $GeoSegment$ by id, as the same segment may appear multiple times in the same route.

We would then add a button labeled Delete GeoSegment, which leads to a dialog allowing a selection of the $GeoSegments$ in the $route$ with a button $confirm$ and $cancel$. When the user clicks $confirm$ the dialog windows checks that the new route is a legal one, meaning checks the invariant, and if it is legal updates the $route$.

If we want to allow multiple deletions at the same time, we must add a function to $Route$ which gets a list of id's, and builds a new route from the old route without these segments.

We would then add a button labeled Delete GeoSegments, which leads to a dialog allowing multiple selections from the $GeoSegments$ in the $route$ with a button $confirm$ and $cancel$. When the user clicks $confirm$ the dialog verifies that the new route would be a legal one, and if it is legal updates the $route$ accordingly.