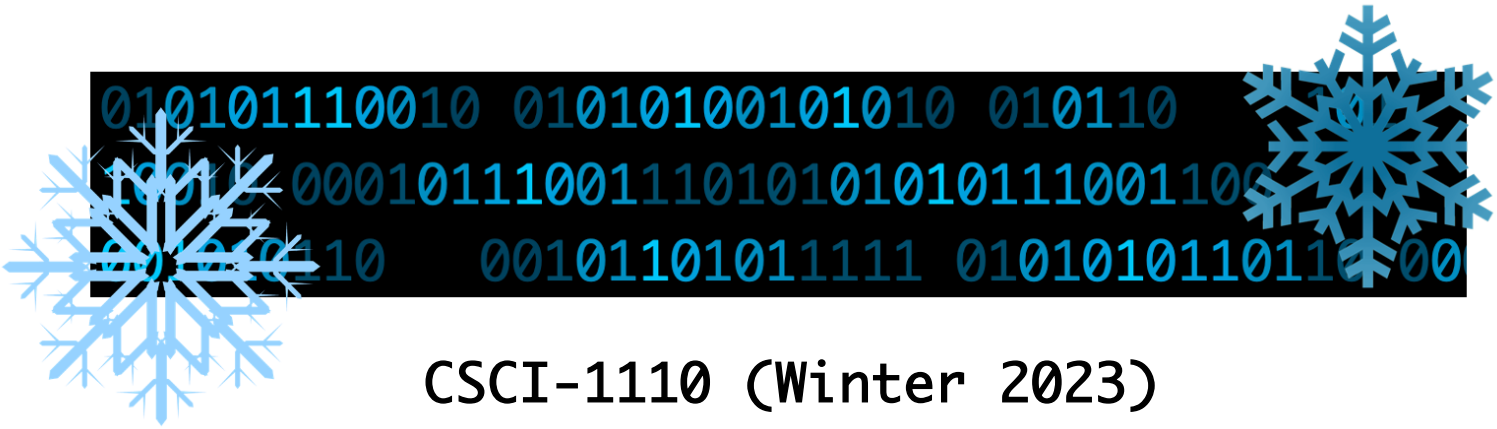


CSCI-1110 (Winter 2023)

ASSIGNMENT 3

The purpose of this assignment is to reinforce your understanding of object-oriented programming and to problem-solve using inheritance, polymorphism, and abstract classes. For this problem you will be provided with sample tests. All input is done via the console and all output is to be done to the console as well. You must submit your assignment via Codio. You can test it by submitting your code.

Note: for this assignment you are expected to install and use an IDE (other than Codio) to develop your code. Please see How-To videos in the How-To Tutorial Section of our Brightspace page on how to install an IDE.

This assignment is divided into three problems, where each problem builds on the next. Each problem subsumes the previous one. Hence, if you complete Problem 3, you will have also completed Problem 2 and Problem 1.

As always, there will be points allocated based on code clarity and code quality. Check out the Code Style Guidelines in Brightspace.

DUE DATE:

This Assignment is due on Monday, March 20th at 14:00 (2PM) Halifax time.

Snow Days of Winter

For this assignment, we are heading to the ski hills! If you've never skied before, there are a few weeks left to get out on the slopes (The hills will last roughly to the due date of this assignment, and then the snow will be gone!).

In this assignment, you will be several aspects of a normal ski hill (SkiHill object). Every ski hill has at least one ski run (modeled by our SkiRun object). Ski runs come in the different forms:

- beginner, green circle (EasyRun) runs;
- intermediate, blue square (MediumRun) runs; and
- more advanced, black diamond (HardRun) runs.

Notably, there are also ◆◆ double-black diamond runs, which are even more fun, but we'll save that challenge for another day!

For a ski hill to be successful, there must be skiers! In problem 3, we'll give you a Skier object and you'll be able to help us finish off our ski hill!

Note: On the last page of this assignment, you'll find the UML diagram for this assignment. It is a great place to start in helping you understand the work ahead.

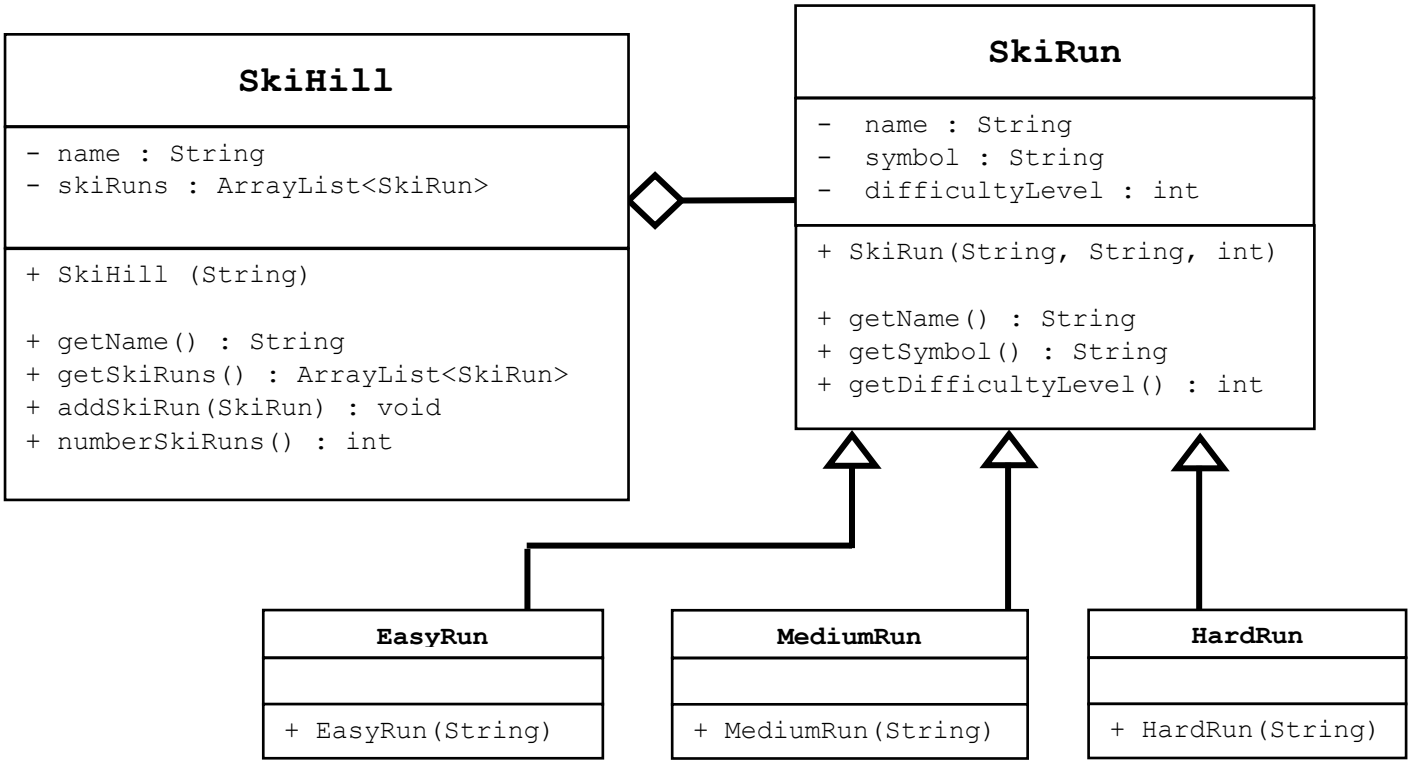
After you've succeeded in building your ski hill (or if you just want some inspiration along the way!), there are a couple of fun ski hills in the area. Martock is about a 45-minute drive from Halifax and a good day out. If you want a few more runs to explore, you can head to Wentworth. It's about 1.5 hours away but a lovely drive. If you do, make sure you say hi to Dr. Siegel! She's there every Saturday and Sunday! ☺

Problem 1 (60%):
SnowDay1 – Build the Hill

For this question, you will receive a few files:

- `SnowDay1.java`
  This contains a functioning main method that you can use to help test your code and generate output. (no work required)
- `SkiHill.java`
  This file represents the SkiHill object.
- `SkiRun.java`
  This file represents the SkiHill object. Every SkiHill has a SkiRun.
- `EasyRun.java`
  This is a type of SkiRun. It is a beginner ski run, represented as a "green circle" symbol.
- `MediumRun.java`
  This is a type of SkiRun. It is an intermediate ski run, represented as a "blue square" symbol.
- `HardRun.java`
  This is a type of SkiRun. It is an advanced ski run, represented as a "black diamond" symbol.

While the UML for the full implementation can be found on the last page, here is the portion of the UML diagram that will be complete after finishing problem 1:



The associated **`SkiHill`** methods act as follows:

- `SkiHill()`
  Constructor method that sets up a ski hill and initializes its `name` (with the given String) and `skiRuns` ArrayList.
- `getName()`
  Returns the String instance variable `name`, which holds the name of the ski hill.
- `getSkiRuns()`
  Returns the instance variable `skiRuns`, which holds an ArrayList of SkiRun objects that represent the ski runs of the given ski hill.
- `addSkiRun(SkiRun)`
  Returns void. It takes in a SkiRun object and adds it to the ski hill by adding it to the instance variable `skiRuns`.
- `numberSkiRuns()`
  Returns the integer representing the number of ski runs the ski hill possesses (i.e. the number of items held in the ArrayList `skiRuns`.

The associated **`SkiRun`** methods act as follows:

- `SkiRun()`
  Constructor method that sets up a ski run and initializes its `name` , `symbol` , and `difficultyLevel` (with the argument values provided).
- `getName()`
  Returns the String instance variable `name`, which holds the name of the ski run.
- `getSymbol()`
  Returns the String instance variable `symbol`, which holds symbol (e.g. “green circle”) that represents thet difficulty level of the ski run.
- `getDifficultyLevel ()`
  Returns the integer instance variable `difficultyLevel`, which holds an integer representing the difficulty level (1=beginner/green circle, 2=intermediate/blue square, 3=advanced/black diamond) of the relevant ski run.

The associated subclasses (EasyRun, MediumRun, and HardRun) of the superclass **`SkiRun`** have constructor methods that act as follows:

- `EasyRun()`
  Constructor method for an EasyRun object that initializes its `name` (based on the string provided), sets the `symbol` to “Green Circle”, and the `difficultyLevel` to 1.
- `MediumRun()`
  Constructor method for a MediumRun object that initializes its `name` (based on the string provided), sets the `symbol` to “Blue Square”, and the `difficultyLevel` to 2.
- `HardRun()`
  Constructor method for a HardRun object that initializes its `name` (based on the string

provided), sets the `symbol` to "Black Diamond", and the `difficultyLevel` to 3.

Once you've created these object classes, all input and output will be handled for you in the `SnowDay1.java` file. You'll know everything is working perfectly on your end when this file runs as expected.

The format of the input that the SnowDay1 file reads in is as follows:

```
Name of the ski hill
Number of ski runs for this ski hill (n)
Run-level Run-name-1
Run-level Run-name-2
...
Run-level Run-name-n
```

and outputs text describing the details of the ski hill. Feel free to play around with your own test input!

Example

Input:

```
Wentworth Southside
6
2 Beaver
2 Embree
2 Gambol
1 Garden Path
3 Giggletree
1 Horse Pastures
```

Output:

```
Welcome to Wentworth Southside!
We hope you enjoy our 6 runs:
1. Beaver (Blue Square - Level 2)
2. Embree (Blue Square - Level 2)
3. Gambol (Blue Square - Level 2)
4. Garden Path (Green Circle - Level 1)
5. Giggletree (Black Diamond - Level 3)
6. Horse Pastures (Green Circle - Level 1)
```

Problem 2 (20%):
SnowDay2 – Close Enough

While we labelled that beginner problem with a Green Circle, the hard part is actually done! We have a functioning ski hill and can now add in some extras!

Before a ski hill opens up its ski runs for the day, they have their Ski Patrol (like safety officers on skis) check out each of the runs to make sure they are safe. The ski patrols look for debris, exposed rocks, and general snow conditions to decide if the run is safe enough to open.

After copying over the files you created in problem 1, you will improve upon that model by implementing a few new methods in both the SkiHill and SkiRun classes to allow for runs to be open or closed so that the ski patrol can do their job!

First, you must add the following instance variable to the **`SkiRun`** object class:

- `runIsOpen : boolean`
  This boolean instance variable will be set to `true` if the fun is open and `false` if the run is closed. Thanks to polymorphism, once you've set this up for your SkiRun class, all subclasses (EasyRun, MediumRun, and HardRun) all have access to this superclass variable as well!

The following methods will be added to the **`SkiRun`** object class:

- `isOpen()`
  Returns the value of the instance variable `runIsOpen`, depicting whether or not the run is open.
- `openRun()`
  Opens the run.
- `closeRun()`
  Closes the run.

The following methods will be added to the **`SkiHill`** object class:

- `openHill()` Opens all ski runs at the hill.
- `closeHill()` Closes all ski runs at the hill.
- `numberOpenRuns()` Returns the number of open runs at the hill.
- `numberClosedRuns()` Returns the number of closed runs at the hill.
- `getOpenRuns()` Returns an ArrayList of type SkiRun containing all open runs at the hill.
- `getClosedRuns()` Returns an ArrayList of type SkiRun containing all open runs at the hill.

Once you've updated these two object classes, all input and output will be handled for you in the `SnowDay2.java` file. You'll know everything is working perfectly on your end when this file runs as expected.

The format of the input that the SnowDay2 file reads input as follows:

```
Same as Problem 1 followed by…
Run-name to close
Run-name to close
...
Run-name to close
CHECK COMPLETE
```

and outputs text describing the details of the ski hill. Feel free to play around with your own test input!

Example

Input:

```
Wentworth Southside
6
2 Beaver
2 Embree
2 Gambol
1 Garden Path
3 Giggletree
1 Horse Pastures
Gambol
Garden Path
CHECK COMPLETE
```

Output:

```
Wentworth Southside is now open!

Ski patrol will make sure that all runs are safe to ski.
CLOSING: Gambol
CLOSING: Garden Path
CHECK COMPLETE

There following 4 runs are open:
Beaver(Blue Square - Level 2)
Embree(Blue Square - Level 2)
Giggletree(Black Diamond - Level 3)
Horse Pastures(Green Circle - Level 1)

The following 2 runs are closed:
Gambol (Blue Square, Level 2)
Garden Path (Green Circle, Level 1)
```

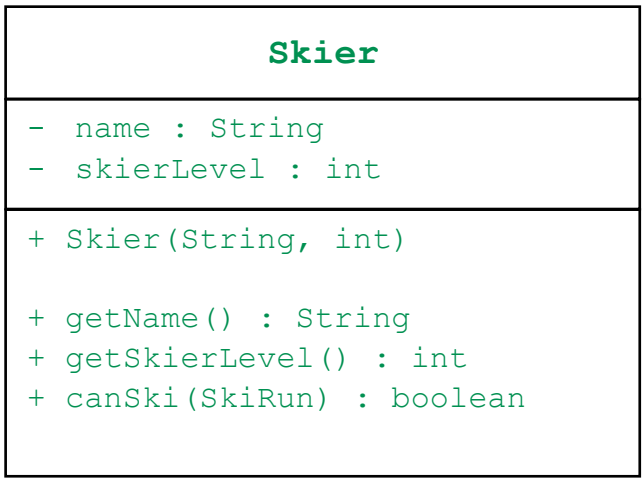
Problem 3 (20%):
SnowDay3 – Let There Be Skiers!

Say hi to Eliana & Dr. Siegel on the slopes!

A ski hill is nothing without skiers! You've done a lot of great work, so enjoy a few runs while we take care of the Skier object class. For this problem, you'll copy over all of your previous object classes. In addition, you'll see two new files: `Skier.java`, the new Skier class we've provided you; and `SnowDay3.java`, the file with which you will be working.

The **`Skier`** class has the following UML diagram and description. Skiers have two properties:

- their `name`, the name of the given skier; and
- their `skierLevel`, an integer from 1 to 3 depicting how good of a skier they are: 1=beginner, 2=intermediate and 3=advanced. This also relates to which hill levels they can ski:
  - A level 1 skier can only ski level 1 runs.
  - A level 2 skier can ski both level 1 and level 2 runs
  - A level 3 skier can ski runs of all levels.



The **`Skier`** class has the following public methods available to you to use:

- `Skier(String, int)`
  Constructor method for the Skier class. It expects the skier name and skier level as input.
- `getName()`
  Returns the name of the skier.
- `getSkierLevel()`
  Returns the skier level.
- `canSki(SkiRun)`
  Returns `true` if the skier is capable of skiing the provided SkiRun (i.e. if the skier level is at least that of the ski run provided), and `false` if the skier level is below that of the run.

And now on to your work! The **`SnowDay3.java`** file is yours to complete. It should expect input as follows:

```
Same as Problem 1 followed by…
Number of skiers (n)
Skier-level Skier-name-1
...
Skier-level Skier-name-n
```

Note: In problem 1, the `SnowDay1.java` main method read in the ski hill and ski run details. You are welcome to copy in any code that makes sense to use!

Your job is to read in these details and make use of the created object classes to output in the following format:

- "Welcome to SkiHill-name!"
- For each of the skiers that are read in, output:
  - "Welcome Skier-name"
  - "Your ski level is: Skier-level"
  - "For your level, check out these runs:"
  - Determine which runs the skier can ski. For each run that the specified skier can ski, output its details as follows:

    "1. Run-name-1 (Run-symbol-1, Level Run-level-1)"
    "2. Run-name-1 (Run-symbol-2, Level Run-level-2)"
    ...
    "n. Run-name-n (Run-symbol-n, Level Run-level-n)"

and outputs text describing the details of the ski hill. Feel free to play around with your own test input!

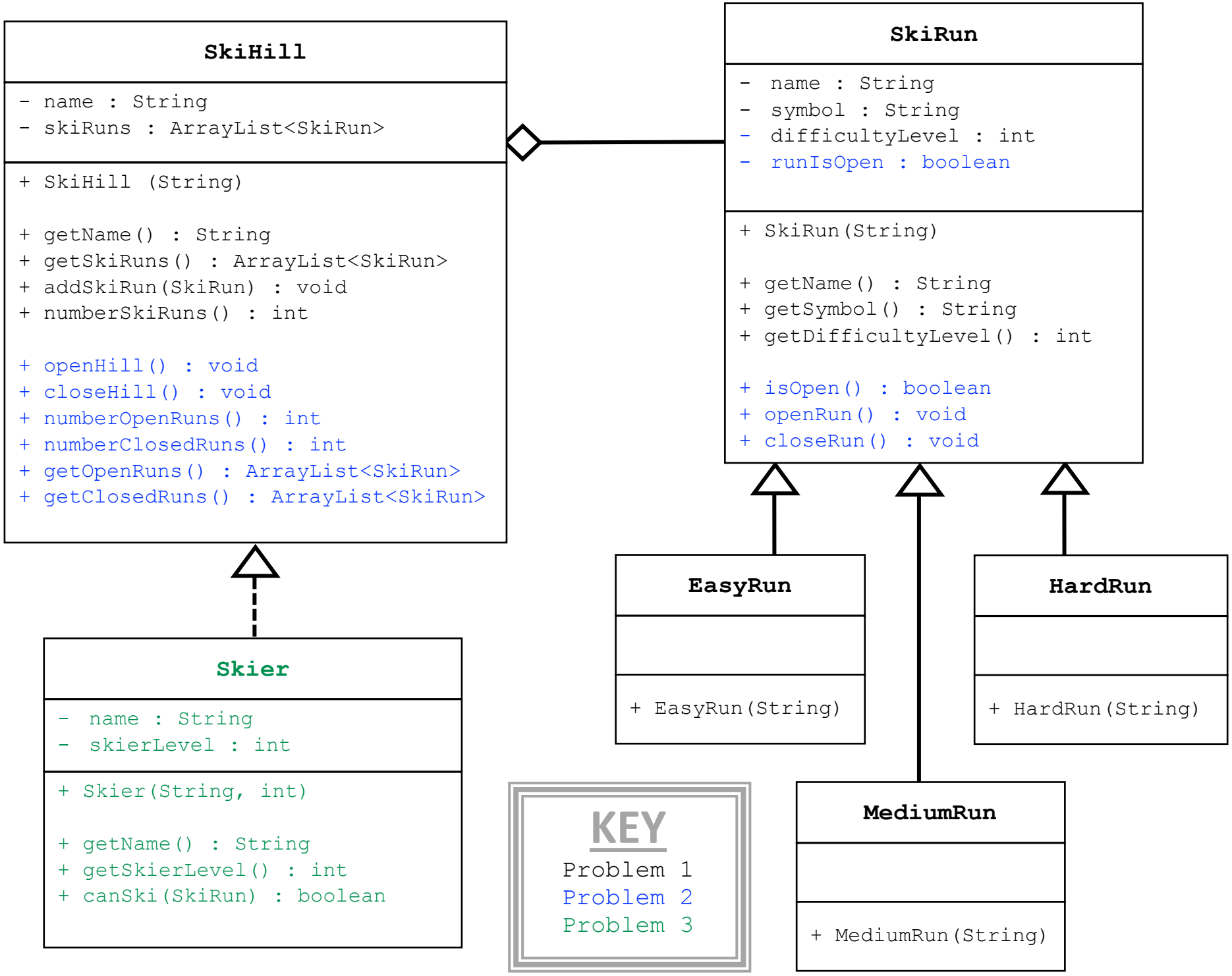
Example

Input:

```
Wentworth Southside
6
2 Beaver
2 Embree
2 Gambol
1 Garden Path
3 Giggletree
1 Horse Pastures
3
1 Natso Fast
2 Medi Umrun
3 Soso Speedy
```

Output:

```
Welcome to Wentworth Southside!

Welcome Natso Fast
Your ski level is: 1

For your level, check out these runs:
1. Garden Path (Green Circle - Level 1)
2. Horse Pastures (Green Circle - Level 1)

Welcome Medi Umrun
Your ski level is: 2

For your level, check out these runs:
1. Beaver (Blue Square - Level 2)
2. Embree (Blue Square - Level 2)
3. Gambol (Blue Square - Level 2)
4. Garden Path (Green Circle - Level 1)
5. Horse Pastures (Green Circle - Level 1)

Welcome Soso Speedy
Your ski level is: 3

For your level, check out these runs:
1. Beaver (Blue Square - Level 2)
2. Embree (Blue Square - Level 2)
3. Gambol (Blue Square - Level 2)
4. Garden Path (Green Circle - Level 1)
5. Giggletree (Black Diamond - Level 3)
6. Horse Pastures (Green Circle - Level 1)
```


SkiHill
- name : String
- skiRuns : ArrayList<SkiRun>
+ SkiHill (String)
+ getName() : String
+ getSkiRuns() : ArrayList<SkiRun>
+ addSkiRun(SkiRun) : void
+ numberSkiRuns() : int
+ openHill() : void
+ closeHill() : void
+ numberOpenRuns() : int
+ numberClosedRuns() : int
+ getOpenRuns() : ArrayList<SkiRun>
+ getClosedRuns() : ArrayList<SkiRun>
SkiRun
- name : String
- symbol : String
- difficultyLevel : int
- runIsOpen : boolean
+ SkiRun(String)
+ getName() : String
+ getSymbol() : String
+ getDifficultyLevel() : int
+ isOpen() : boolean
+ openRun() : void
+ closeRun() : void
Skier
- name : String
- skierLevel : int
+ Skier(String, int)
+ getName() : String
+ getSkierLevel() : int
+ canSki(SkiRun) : boolean
EasyRun
+ EasyRun(String)
HardRun
+ HardRun(String)
MediumRun
+ MediumRun(String)
KEY
Problem 1
Problem 2
Problem 3