Please read the instructions thoroughly then create a program to solve the problem.

Note:

* For your solution, we request that you use C#, Java, Kotlin, Python, Typescript/Javascript.
* Your application must run.
* You should provide sufficient evidence that your solution is complete by indicating that it works correctly against the supplied test data (eg. a visual screenshot to accompany the submission results).
* Please use the instructions at the bottom of this email to submit your code. Here are a few tips:
  + Include unit tests in your solution. We recommend using TDD to solve these problems, where possible.
  + Keep your methods and classes small.
  + Demonstrate a good understanding of clean code, object oriented skills, and SOLID principles.
  + Submit production-ready code that is clean and easy to understand with ample documentation.

Rules:

1. You may NOT use any external libraries to solve this problem, but you may use external libraries or tools for building or testing purposes. Specifically, you may use unit-testing libraries or build tools available for your chosen language (e.g., JUnit, Ant, NUnit, Pytest, Jest, etc.).
2. You should NOT include any executable attachments, including those with .exe or .lib extensions. System security is very important to us and certain file extensions will be blocked for security purposes, resulting in delays to your application. We need to be able to run and build your code ourselves, so please submit your code as a zipped file of source code and supporting files, without any compiled code. If you're submitting in C#, please do not submit your code as a .msi file.
3. Please compress your files into a single .zip file before upload. Kindly ensure there are no executables in your submission. Our system blocks executable files for security purposes, and we want to avoid any delays in your process. (Executables include asp, bat, class, cmd, com, cpl, dll, exe, jar, jsp, jse, pif,w scr, shs, sh, vb, vbe, vbs, ws, wsc, wsf, wsh & msi)
4. Please include a brief explanation of your design and assumptions, along with your code, as well as detailed instructions to run your application. Also include the environment required to run the application, eg. Windows, Linux.
5. We assess a number of things including the design aspect of your solution and your object oriented programming skills. While these are relatively small problems, we expect you to submit what you believe is production-quality code; code that you'd be able to run, maintain, and evolve. You don't need to gold plate your solution, however we are looking for something more than a bare-bones algorithm.
6. At GIC, we highly value integrity. We also want our hiring process to be fair, and for everyone to start from the same place. To enable this, we strongly request that you do not share or publish these problems.

**Bank Account**

You're designing a simple banking system that handles operations on bank accounts. At the moment, your system is capable of three features:

* depositing an amount
* withdrawing an amount
* printing account statement

User input and output can be from command line.

When account is created, its balance is 0.

When launching the application, it prompts user for actions:

Welcome to AwesomeGIC Bank! What would you like to do?

[D]eposit

[W]ithdraw

[P]rint statement

[Q]uit

User should be able to enter D or d to select Deposit option. Similarly, initial character is used for other options.

**Deposit**

Upon selecting Deposit option, application prompts user for amount.

Please enter the amount to deposit:

User is then able to enter:

500

Then system responds with:

Thank you. $500.00 has been deposited to your account.

Is there anything else you'd like to do?

[D]eposit

[W]ithdraw

[P]rint statement

[Q]uit

**Withdraw**

User can choose to withdraw by entering W/w.

w

System then responds with:

Please enter the amount to withdraw:

User is able to enter:

100

Then system responds with:

Thank you. $100.00 has been withdrawn.

Is there anything else you'd like to do?

[D]eposit

[W]ithdraw

[P]rint statement

[Q]uit

You can ignore where the withdrawn amount goes for now.

**Print Statement**

When user chooses to print statement, user enters

p

System then responds with:

Date | Amount | Balance

8 Jul 2022 11:12:30AM | 500.00 | 500.00

8 Jul 2022 11:14:15AM | -100.00 | 400.00

**Quit**

When user chooses to quit, user enters:

q

System responds with:

Thank you for banking with AwesomeGIC Bank.

Have a nice day!