Technical Tutorial

Quantum Computing with Strange and Maven

Strange is a quantum computing platform for Java.

Prerequisites: ***Maven***, JDK 21, git, eclipse and some Java experience.

Last update: November, 2024

Use a Maven Project to Build and Run a Quantum Program

1. Navigate to https://www.manning.com/books/quantum-computing-in-action.
2. Download the source code.
3. Extract the contents of the archive to your Desktop.
4. Using the cmd line, navigate to the project in Chapter 2.
5. After reviewing the files, build and run the project using Maven: mvn clean javafx:run
6. Save the file with the Java code (Main.java) to your Desktop for use in a later step.

Build the Strange Library from Source

1. `cd ~\Desktop`
2. Download the source from github.
   `git clone https://github.com/redfx-quantum/strange.git`
3. Build the JAR file: mvn install
4. Copy the JAR file to your Desktop.
5. Remove or comment the package statement from the Main.java file on your Desktop.
6. Run the program:
   `java -cp strange-0.2.0-SNAPSHOT.jar Main.java`

Run the Quantum Program with eclipse

1. Create a new Java Project: File; New; Java Project and name it “random”
2. There is no need for a Module.
3. Create a “lib” folder within your project (at the same level as “src”).
4. Copy the Strange JAR created above into your new lib folder.
5. Right click on the JAR file and add it to your Build Path.
6. Copy the Main.java file from your Desktop to the project’s src folder.
7. Build and run your project.

Use the Quantum High-level API to Create a Random int

1. Modify your working program, from above, that creates a random bit in order to create a random integer of any size.
2. Make it easy to specify the size of the int in bits (e.g., use a hard-coded int to specify the number of bits). Alternatively, you could allow the end-user to specify the number of bits from the command line.
3. Use a Quantum Computer (or simulator) to generate that number of quantum bits.
4. Convert those random quantum bits to a single integer. For example:
   a. Given 4 bits as input, call randomBit four times.
   b. Assume that the QC generated: 1, 1, 0, 1
   c. Display 13 (since 1101 in binary is 13 in decimal).
5. Finally, display the random value as a hexidecimal number instead of decimal.

NOTE: Cryptographic systems, like RSA, often require very large integers (e.g., 3072 bits) to make their keys more secure. These large integers are commonly displayed in hex since the string of digits is shorter.