Describe the round trip Java RMI run-time processing of the following method:

```
Result foo( int aNum, String aName)
```

Assume the method is in remote object `target` located in machine “server” and is invoked by a calling method in a physically separate machine “client”. Note any assumptions you are making in describing the execution of the remote service. You may draw a diagram if you like.

```
// i.e. client side method makes the following call:

Result myResult = target.foo( 24, "Hello Hello");
```

Assumptions:
1. `target` is registered as a remote object in a well known RMI registry.
2. A stub as been created for object `target`
3. A remote interface exists for `target` and the return object `Result`.

Sequence (all success) of actions:
1. client binds to the RMI registry and obtains a remote reference for object `target`.
2. call is made to `foo()` method using stub for target object.
3. target stub marshals the calling parameters: 24 is passed by value, a copy of the string “Hello Hello” is created and serialized.
4. a message is created to be transported to the location specified by the remote reference for target.
5. RMI assigns an ID to the remote call and starts a timer.
6. the message travels across the network and arrives at server.
7. an RMI dispatcher service receives the message at the server and unmarshalls the parameters.
8. the dispatcher looks up the local object based on the remote reference and uses Java reflection to invoke the `foo()` method with the passed parameters.
9. `foo()` does its thing and creates a return object (Result), control is passed back to the dispatcher.
10. the dispatcher marshals the Result object using serialization.
11. a message is created to be transported to the client location
12. the message travels across the network and arrives at the client.
13. the stub for target receives the message at the client and unmarshalls the return object Result. (note that Result could have been a known remote object or the Result class could have been dynamically loaded as part of the return)
14. normal Java execution returns control to the point where the initial call was made.