Java RMI example:
Simple remote calculator service

PAYAP UNIVERSITY

Exporting Remote Objects

- Implementing a remote interface is not enough: to accept remote calls a remote object must be exported
  - Basically, made to listen on a socket for incoming calls
  - Done automatically in the constructor, if the remote object subclasses from `java.rmi.server.RemoteObject` (typically through `UnicastRemoteObject`)
  - Alternately, done with the static method `UnicastRemoteObject.exportObject`

- All constructors of the remote object must throw a `RemoteException`

PAYAP UNIVERSITY

Parameters in RMI

- pass-by-value
  - primitive types
  - object parameters (*object serialization*)

- pass-by-reference
  - remote objects

PAYAP UNIVERSITY

Security Manager

- The SecurityManager defines what downloaded code is allowed to do; by default, not much
- If code will be downloaded, the SecurityManager must be defined

```
if (System.getSecurityManager() == null) {
    System.out.println("security manager being constructed");
    System.setSecurityManager(new RMISecurityManager());
  }
```

PAYAP UNIVERSITY

Security Manager: Policy files

- Both Client and Server must specify a security policy
- Allows downloaded code to
  - Accept connections on any user port
  - Connect to any http port
- Permission entries
  - FilePermission, RuntimePermission, etc.

```
grant {
    permission java.net.SocketPermission "*:1024-65535",
        "connect,accept";
    permission java.net.SocketPermission "*:80", "connect";
};
```

Distributing RMI software

- Important system properties:

```
-Djava.security.policy=./policy.all
-Djava.rmi.server.hostname=cis.payap.ac.th
-Djava.rmi.server.codebase=http://cis.payap.ac.th/classes
```