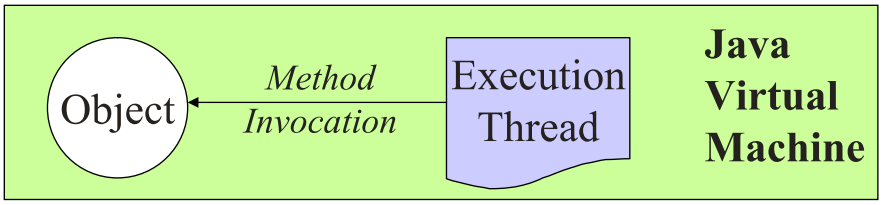
Java RMI : Remote Method Invocation

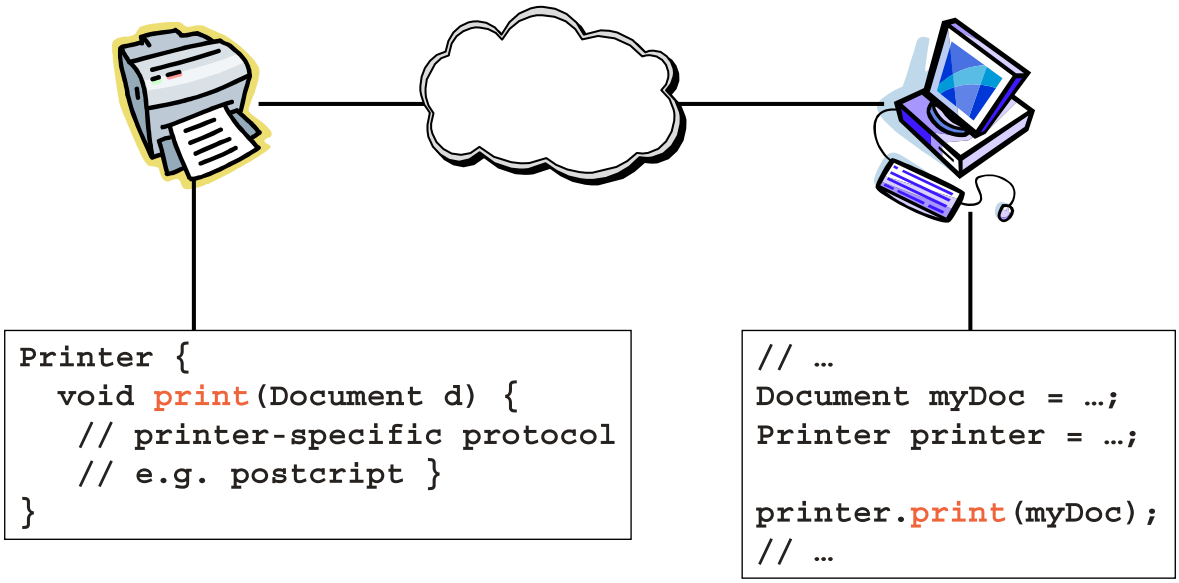
Introduction : Remote Computation

- Objects encapsulate **data** + **operations**
- Usually stored and evaluated **locally**



- **Remote** storage/evaluation can also be useful :
  - Object encapsulates physical resource (e.g. Printer)
  - Data resides remotely and is very large (e.g. phone directory lookup)

Example: Print Object



```
Printer {
  void print(Document d) {
   // printer-specific protocol
   // e.g. postcript }
}
```

```
// …
Document myDoc = …;
Printer printer = …;

printer.print(myDoc);
// …
```


Remote Print Object Implementation

- How can Java support remote operations ?
- Use of **proxy** objects :

```
public class Printer_Proxy {
  public void print(Document doc){
    // magic communication
} }
```

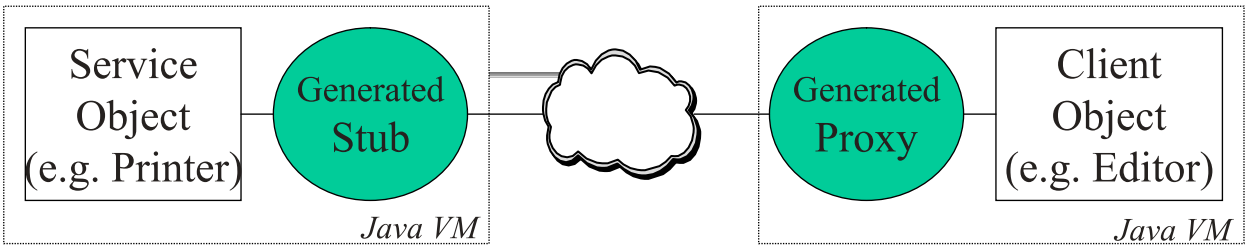
```
Document myDoc = …;
Printer printer = …;

printer.print(myDoc);
```

Java VM

Remote Method Invocation Overview

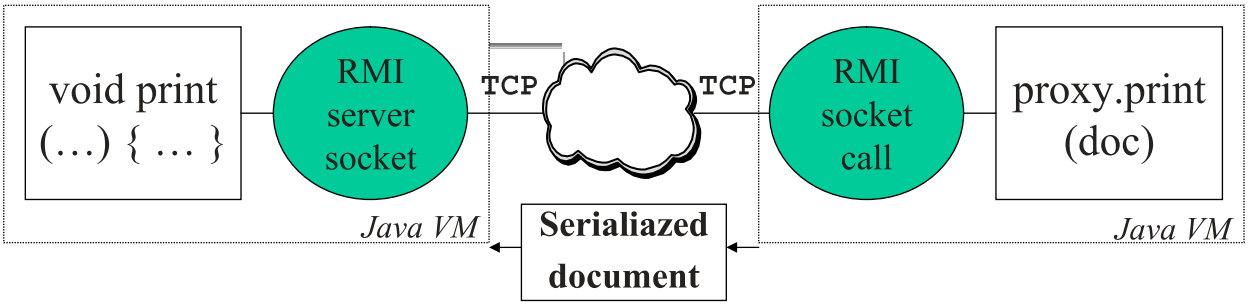
- RMI is Java's mechanism for **automatically** generating proxy classes.
- User codes service and client objects
- RMI compiler generates network communication code



Remote Interface Example

- What ties it all together ?
- *Answer*: Server, stub, proxy and client all share the same **remote interface**

```
public interface PrintService extends java.rmi.Remote {
   public void print(Object obj)
       throws java.rmi.RemoteException;
}
```



RMI Features

- Language specific (Java)
- Object oriented
  - Full objects as parametears
  - Supports design patterns
- Mobile behavior
  - Move interface implementation from client to server, and server to client
- Safe & Secure (Java VM security)
- Connects to existing/legacy (JNI/JDBC)

RPC versus RMI

- Procedural
- Language Independent
- External data representation (XDR)
- Basic types as parameters
- Pointers require explicit handling
- No code mobility (same for CORBA, DCOM)

- Object Oriented
- Language Specific
- Java Object Serialization
- Any object implementing serialization as parameter
- References to local and remote objects handled automatically (deep copy)
- Mobile code (Java byte-code)

RMI Terminology

- A ***remote object*** is one whose methods can be invoked from another Java Virtual Machine, potentially on a different host.
- ***Remote method invocation*** (RMI) is the action of invoking a method of a remote interface on a remote object.

```
// Local method invocation example
HashTable table = new HashTable();
table.put("akonstan", "secRet!");
```

```
// Remote method invocation example (incomplete)
PasswordDb db = (PasswordDb) Naming.lookup("//myhost/cs4119db");
db.put("akonstan", "secRet!");
```

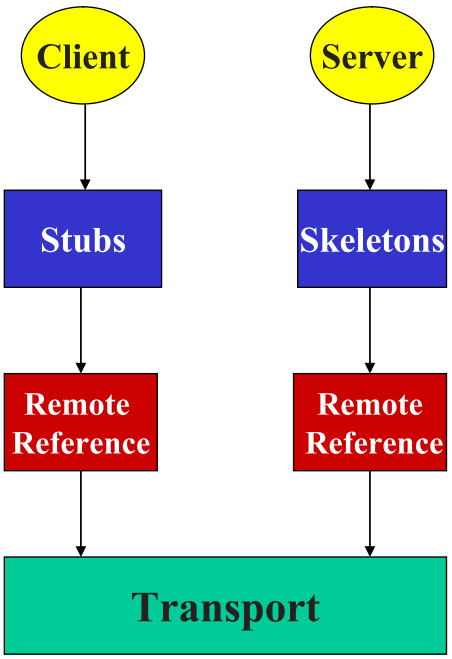

Remote Invocation Semantics

The semantics of remote method invocations differ in some ways from those of local method invocations :

- Clients interact with remote ***interfaces***.
- Non-remote arguments, and results from, a remote method invocation are passed by ***copy*** rather than by reference.
- A remote object is passed by ***reference***, not by copying the actual remote implementation.
- Clients invoking remote objects must handle ***additional failure modes*** (exceptions)

Java RMI Architecture

- Servers extend `RemoteObject` and **implement** remote interfaces.
- Any serializable object can be sent as a parameter or returned as a response
- The RMI compiler generates client stubs (proxies) and server skeletons(dispatchers)



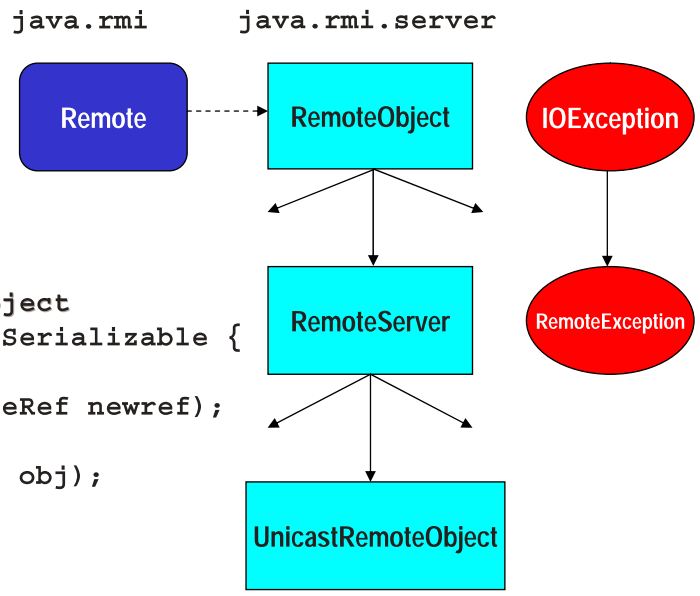
Java Object Serialization

- RMI parameters passed as serialized objects
- Serialized objects are converted to a **stream of bytes**.
- Serialization stores the class structure along with the values of the object (class structure only stored once per class).
- Serialization handles **references** by traversing them and serializing objects along the way.
- You do not need to write any special code to utilize the serialization routines. It is sufficient to implement the `java.io.Serializable` interface (this is a marker interface and does not define any methods).

RMI Interfaces and Classes

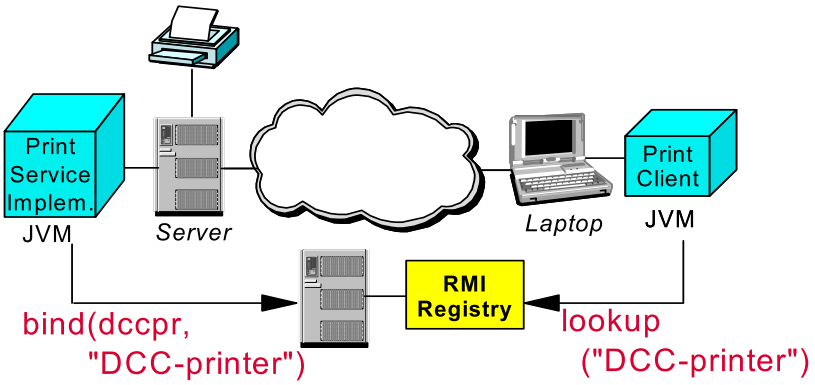
```
public interface Remote {}
```

```
public abstract class RemoteObject
   implements Remote, java.io.Serializable {
  protected RemoteObject();
  protected RemoteObject(RemoteRef newref);
  public int hashCode();
  public boolean equals(Object obj);
  public String toString();
}
```



Locating servers with RMI Registry

- RMI registry is the object directory service.
- Objects are bound to the registry using string names.
- The registry process may execute on any network host.
- RMI URL: `rmi://myhost:1099/DCC-printer`



RMI Example