

CITS3213: CONCURRENT PROGRAMMING

Java RMI and CORBA

- Java RMI and CORBA both support programming with distributed objects.
- Both allow a programmer to treat remote objects like local ones when making method calls, storing objects in data structures, etc., by using stub/proxy objects.
- What are the advantages and disadvantages of each? Are there situations where one should be preferred?
- The differences between them mostly stem from a single fundamental difference: Java RMI is specific to Java, while CORBA is language independent.

Java RMI vs CORBA: Language Dependence

- The main advantage of CORBA is that it allows a system to work with objects written on many different languages.
- This is particularly important for interfacing to existing systems, which could be written in any language.
- It is also important for considering possible future access to a system from other languages.
- Java RMI has the advantage of being simpler: interfaces can be defined directly in Java, as can inheritance, exceptions, etc.
- Java RMI also has the advantage of being able to easily support features like creating local copies of remote objects and distributed garbage collection that may not fit well with some languages.

RMI: Passing Objects by Remote Reference or By Local Value

```
// Will be passed by reference
public interface Hello extends java.rmi.Remote{
    String sayHello() throws java.rmi.RemoteException;
}

public interface HelloReceiver extends java.rmi.Remote{
    String setHello(Hello hello) throws java.rmi.RemoteException;
}

// Will be passed by value (local copy made by receiver)
public interface HelloBV extends java.io.Serializable {
    String sayHello();
}

public interface HelloReceiverBV extends java.rmi.Remote{
    String sayHello(HelloBV hello) throws java.rmi.RemoteException;
}
```

Java RMI vs CORBA: Summary

Java RMI	CORBA
Java only	Language independent Can access foreign language objects
Interfaces specified in Java	Interfaces specified externally in IDL
Pass objects by remote reference or by value (making local copy)	Pass objects by reference (by value possible but awkward)
Distributed garbage collection integrated with local collectors	Remote references manually released
Generally simpler to use	More complicated