Arguments and Results in RMI

- Semantics of passing arguments for RMI in object-oriented languages needs to be defined. Why?
- Argument and Result passing in Java RMI:
  - When type of parameter is defined as remote interface, argument or result is passed as ROID.
  - Other non-remote objects may be passed *by value* if they are *serializable*.
- Which objects can be accessed by RMI?
  - Any object can be accessed by RMI
  - Distinguish between remote objects and local objects. (e.g. keywords or classes with interface compiler)
  - Use interface definition language (IDL)
- Problem: migration/replication

Location of Objects

- Is location of objects an issue?
- Mapping from name to ROID.
  - e.g. `lookup(`*`name`*`)` -> proxy with ROID
- Mapping from ROID to location.

Dynamic Binding

- Dynamic method binding should also apply to RMI.
- Smalltalk: Allow any message to be sent to any object, and raise exception if method is not supported.
  - Distributed Smalltalk: general-purpose proxies.
- Java RMI:
  - dynamic binding as a natural extension of local case
  - Example:

```
Shape aShape = (Shape) stack.pop();
float f = aShape.perimeter();
```

Garbage Collection

- Some languages (Java, Smalltalk) support garbage collection.
- Explicit memory management difficult/impossible in distributed environment.
- Distributed garbage collection typically realized in ROID modules. Each ROID module:
  - keeps track how many sites hold remote ROIDs for each local object
  - informs other ROID modules about generation/deletion of ROIDs for their local objects.
- Local garbage collector collects objects with no local or remote references.
- Reference counting (`addROID()`/`removeROID()`) over unreliable networks?