Java MPI Quick Reference I

MPI, the **Message Passing Interface**, is a standardized message-passing system for writing portable parallel programs.
Our implementation: MPJ Express

run programs: mpjrun.bat <mpjrun arguments> <jar file> <program arguments>
- requires a machines file in current directory with ip addresses of hosts
- mpjrun arguments: `C` or `-np 4`

process ID: **rank** (0, 1, 2, ...)

`MPI.Init`()
`MPI.Finalize()`
`Comm.Rank`()
`Comm.Size`()

message envelop: source, destination, tag (message type) and communicator
- `MPI.COMM_WORLD` = `Comm` object with all processes

Point-to-point communication: `Comm.Send` & `Comm.Recv`
- messages are non-overtaking, but fairness is not guaranteed
- types must match!
- `Comm.Recv`: - count is upperbound
  - use `MPI.ANY_TAG`, `MPI.ANY_SOURCE`
- `Comm.Probe`, `Comm.IProbe` (non-blocking): polling for messages
- datatypes: `MPI.INT`, `MPI.CHAR`, `MPI.FLOAT`, `MPI.DOUBLE`, ...

Varia
- `Status`: object with fields `source` and `tag`. Get count with `Get_elements`
- start processes `MPI_Comm_spawn`
- throw `MPIException` when error

Communication optimization:
- **standard**:send immediately or big messages: buffered & blocked
- **non-blocking:** post > comm. in background > test-for-completion
  - send buffer may not be accessed!
- **modes**
  - **buffered (B)**: message is copied (MPI_Buffer_attach to specify buffer)
  - **synchronous (S)**: rendez-vous of sender & receiver
  - **ready-mode (R)**: receiver is ready => sender can send immediately
- MPI_Cancel: cancellation of non-blocking communications

posting: `Comm.Isend` and `Comm.Irecv`()
test: `Comm.Wait`(), `Comm.Test`() , `Comm.Request_free`()

blocking	*non-blocking*
`Comm.Bsend`	`Comm.Ibsend`
`Comm.Ssend`	`Comm.Issend`
`Comm.Rsend`	`Comm.Irsend`

`Comm.Send_recv` & `Comm.Send_recv_replace` (same buffer) (both blocking)
- use less memory & avoid deadlock