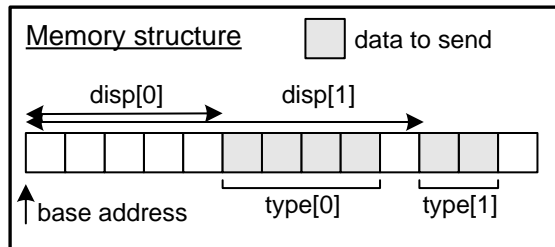# Java MPI Quick Reference II

User-defined datatypes (`DataType` object):

for non-contiguous memory locations
Typemap = {(type$_0$, displacement$_0$), ...}
specifies message buffer together with base address (passed
 with send/receive command)

Memory structure — data to send

disp[0]   disp[1]

base address   type[0]   type[1]

Create types with (and commit with `Commit` command!)
`Contiguous`: replication of a datatype
`Vector`: equally spaced blocks (stride)
`Hvector` stride is arbitrary nbr of bytes
`Indexed` different #blocks, # displacements
`Hindexed`
`Struct` diff datatypes (most general)

pack/unpack alternative
make a copy to a buffer: but consumes more memory and cycles.
E.g. when buffer layout is data dependent or for sending objects
Use inherent serialization capacities of java to write objects to a
bytestream which will then be send.
Do this instead of using the Pack & Unpack features of MPI for C,
which has no inherent serialization.

Error handling

an `MPIException` is thrown when an error occurs

Communicators

> subdivide processes into **groups** (type MPI_GROUP)
process has (different) rank in each group it belongs to
MPI_Group_rank: MPI_UNDEFINED if no member
MPI_Group_size
MPI_Comm_group: get group of communicator
 ...

> **communicator** is asociated with one or two groups
communication only happens within a communicator
intracommunicator: within a group
intercommunicator: P2P between 2 disjoint groups
`MPI.Comm_world` = communicator of all processes

Collective Communications (`Intracomm` object)

`Barrier`: returns when all processes entered the call
`Bcast`: broadcast (one-message-to-all), root also!
`Gather`: all-to-one (root included)
`Gatherv`: with stride
`Scatter`: diff-messages-to-all (`Scatterv`)
`Allgather`: all-to-all (`Allgatherv`)
`Alltoall`: personalized messages all-to-all
`Alltoallv`
`Reduce`: global reduce operation all-to-one
 Operations (`Op` object): `Max`, `Min`, `Sum`, `Prod`, ...
`Allreduce`: result to all
`Reduce_scatter`: scatter the result (array)
`Scan`