```
// import mmt terms
import info.kwarc.mmt.api.Path
import info.kwarc.mmt.api.uom.OMLiteral._
import info.kwarc.mmt.api.objects._

// create a group inside of MMT
val mmt_term = OMA(
 OMS(Path.parseS("http://www.gap-system.org?pcgroup1?pcgroup_by_pcgscode")),
 List(
   OMI(BigInt("114408488571536161623939587401849792853027787 17")),
   OMI(512)
 )
)

/**
 * (http://www.gap-system.org?pcgroup1?pcgroup_by_pcgscode
114408488571536161623939587401849792853027787 17 512)
 */
println(mmt_term)



// encode it into an OpenMath term (for GAP in this case)
import info.kwarc.mmt.odk.OpenMath.Coding.GAPEncoding
val om_term = GAPEncoding.decodeExpression(mmt_term)

/**
 * OMApplication(
 *   OMSymbol(pcgroup_by_pcgscode,pcgroup1,None,None),
 *   List(
 *     OMInteger(114408488571536161623939587401849792853027787 17,None),
 *     OMInteger(512,None)
 *   ),None,None)
 */
println(om_term)

// prepare a computation for GAP
// here we compute the nr of conjugacy classes
import info.kwarc.mmt.odk.OpenMath._
val NrConjugacyClasses = OMSymbol("NrConjugacyClasses", "scscp_transient_1", None, None)
val computation = OMApplication(NrConjugacyClasses, List(om_term), None, None)

// fetch the resulting expression from GAP
val client = SCSCPClient("scscp.gap-system.org")
val om_result = client(computation).fetchExpression()
client.quit()

/**
 * OMInteger(92,None)
 */
```

```
println(om_result)

// and turn the result back into an MMT term
val mmt_result = GAPEncoding.encode(om_result)

/**
 * 92
 */
println(mmt_result)
```