How to extend the interface with a new parameter or force:

1. How to add a new parameter in ReaDDy?
   See tutorial_PotentialAddingInReaDDy.pdf
2. How to add this parameter to ReaDDyMM?
   In the TopMM.java file are all parameters stored in an array, to be than handed over to OpenMM. For the new parameter you have to add a new case-clause in either the order one or order two potential section.

```
case "your_parameter_name": {
   potParam1.add(123.0);        // first value => unique number for your
                                // parameter (<1000)
   potParam1.add(Double.parseDouble(entry.getValue())); // following: value(s)
   break;
}
```

   If the amount of transduced values is unclear at this time, it is important, to submit the amount of parameters too.
   After storing your parameter, you need to interpret it in the C++-interface. In the C++ code you add a new case-clause in either the order one potential(external force) or order two potential (pairwise force).

```
else if(paramPot1[i]==your parameter number){
    i++;
     for( int x=0, x<howMuchValues, ++x){ // iterate through your
          // parameter-values and store them
         myParam=paramPot1[i];
         i++;
     }
}
```

   The i indicates the current position in the array. Don't forget to increase it properly after reading your variable.
   Now you can use your new parameter in a force in OpenMM

3. How to add a new force to ReaDDy?
   See tutorial_PotentialAddingInReaDDy.pdf
4. How to add a new force to ReaDDyMM?
   In the TopMM.java add a new case-clause in the "type"section of either the order one or two potentials.

```
case "YOUR_POTENTIAL_NAME": {
    potParam1.add((int)uniquePotentialID); //(>1000)
    break;
}
```

   Invent an unique ID for your new potential, and be sure, that it is unique and bigger than 1000.
   ReaDDy will store all related parameters for this potential in the respective array. You now need to specify a OpenMM force-function in the C++-library.

```
if(type==1234){          // your unique potential ID
      ss << "O*0.5*" << K << "*((" << center[2] << "+z)^2 +
min(200-sqrt((x-" << center[0]<<")^2 + (y-" << center[1] << ")^2),0)^2)" ;
      // ss is a string stream from witch contains your force formula
      // you can use here your various parameters
      // there are two predefined parameter
      useO = true;      // parameter "O" witch is important for activating
                        // deactivating particles
      useR = true;      // and R -  the particle radius
      // you can decide whether to use them by switching this booleans on
      // or off
      // add here possible per particle parameter
  }
```

   The „O“ parameter is important for the activation and inactivation of particles. When a particle changes its type through a reaction, it is necessary, to switch off all to it applied forces. This is done through this

parameter. The parameter R, the particle radius, is a common used parameter in force formulas. If you want to add another per particle parameter, you have to add the following code into the above if-clause.