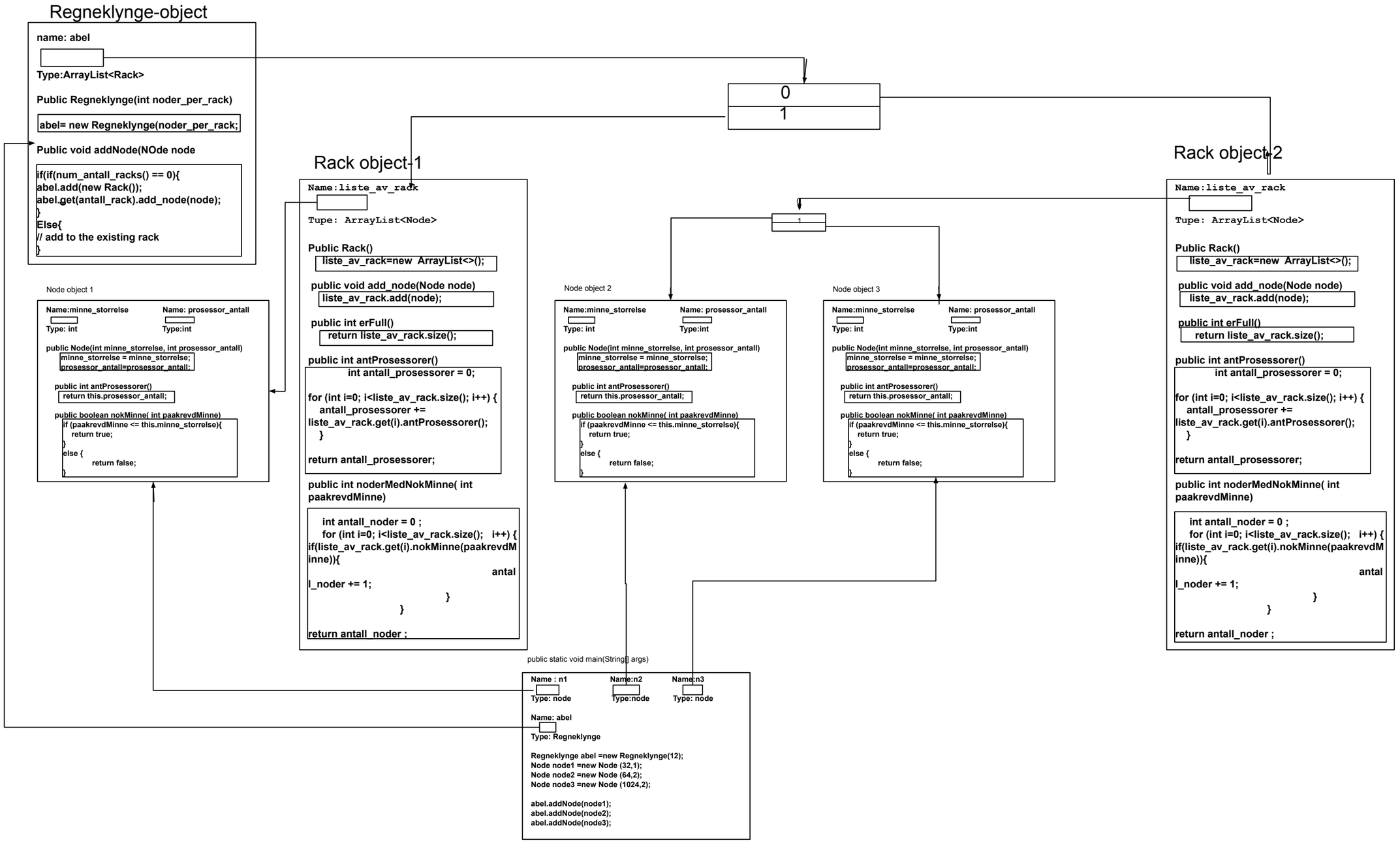
Regneklynge-object
name: abel
Type:ArrayList<Rack>
Public Regneklynge(int noder_per_rack)
abel= new Regneklynge(noder_per_rack;
Public void addNode(NOde node
if(if(num_antall_racks() == 0){
abel.add(new Rack());
abel.get(antall_rack).add_node(node);
}
Else{
// add to the existing rack
}
0
1
Rack object-1
Name:liste_av_rack
Tupe: ArrayList<Node>
Public Rack()
liste_av_rack=new ArrayList<>();
public void add_node(Node node)
liste_av_rack.add(node);
public int erFull()
return liste_av_rack.size();
public int antProsessorer()
int antall_prosessorer = 0;
for (int i=0; i<liste_av_rack.size(); i++) {
antall_prosessorer +=
liste_av_rack.get(i).antProsessorer();
}
return antall_prosessorer;
public int noderMedNokMinne(int
paakrevdMinne)
int antall_noder = 0 ;
for (int i=0; i<liste_av_rack.size(); i++) {
if(liste_av_rack.get(i).nokMinne(paakrevdM
inne)){
antal
l_noder += 1;
}
}
return antall_noder ;
Rack object-2
Name:liste_av_rack
Tupe: ArrayList<Node>
Public Rack()
liste_av_rack=new ArrayList<>();
public void add_node(Node node)
liste_av_rack.add(node);
public int erFull()
return liste_av_rack.size();
public int antProsessorer()
int antall_prosessorer = 0;
for (int i=0; i<liste_av_rack.size(); i++) {
antall_prosessorer +=
liste_av_rack.get(i).antProsessorer();
}
return antall_prosessorer;
public int noderMedNokMinne(int
paakrevdMinne)
int antall_noder = 0 ;
for (int i=0; i<liste_av_rack.size(); i++) {
if(liste_av_rack.get(i).nokMinne(paakrevdM
inne)){
antal
l_noder += 1;
}
}
return antall_noder ;
1
Node object 1
Name:minne_storrelse
Type: int
Name: prosessor_antall
Type:int
public Node(int minne_storrelse, int prosessor_antall)
minne_storrelse = minne_storrelse;
prosessor_antall=prosessor_antall;
public int antProsessorer()
return this.prosessor_antall;
public boolean nokMinne(int paakrevdMinne)
if (paakrevdMinne <= this.minne_storrelse){
return true;
}
else {
return false;
}
Node object 2
Name:minne_storrelse
Type: int
Name: prosessor_antall
Type:int
public Node(int minne_storrelse, int prosessor_antall)
minne_storrelse = minne_storrelse;
prosessor_antall=prosessor_antall;
public int antProsessorer()
return this.prosessor_antall;
public boolean nokMinne(int paakrevdMinne)
if (paakrevdMinne <= this.minne_storrelse){
return true;
}
else {
return false;
}
Node object 3
Name:minne_storrelse
Type: int
Name: prosessor_antall
Type:int
public Node(int minne_storrelse, int prosessor_antall)
minne_storrelse = minne_storrelse;
prosessor_antall=prosessor_antall;
public int antProsessorer()
return this.prosessor_antall;
public boolean nokMinne(int paakrevdMinne)
if (paakrevdMinne <= this.minne_storrelse){
return true;
}
else {
return false;
}
public static void main(String[] args)
Name : n1
Type: node
Name:n2
Type:node
Name:n3
Type: node
Name: abel
Type: Regneklynge
Regneklynge abel =new Regneklynge(12);
Node node1 =new Node (32,1);
Node node2 =new Node (64,2);
Node node3 =new Node (1024,2);
abel.addNode(node1);
abel.addNode(node2);
abel.addNode(node3);