**Example1**

功能：创建一个含有三个Machine的网格资源（每个Machine可以包含一个或多个PE）

注意：在创建任何GridSim实体前，初始化GridSim包

程序结构：

**public** **class** Test1 {

**public** **static** **void** main(String[] args) {

* + - 1. 在控制台打印开始和结束信息；
      2. 初始化GridSim包；
      3. 调用createGridResource()方法创建网格资源；

}

**private static** GridResource createGridResource(){

1. 创建机器列表，存储一个或多个机器（MachineList，继承LinkedList<Machine>）；
2. 创建一个机器，并将其添加至机器列表（Machine(3个参数)）；
3. 若需要创建更多机器，重复第二步（代码一样！）；
4. 创建一个资源属性对象（ResourceCharacteristics），存储网格资源特性；
5. 创建一个网格资源对象（需要捕获异常，防止未初始化GridSim包，或实体名为空，或PE数为0）；
6. 返回一个网格资源；

}

}

**Example2**

功能：创建网格任务，并打印任务列表

说明：本例不需要初始化GridSim和SimJava，因为创建任务与运行模拟不相关

程序结构：

**public** **class** Test2 {

**public** **static** **void** main(String[] args) {

* 1. 调用createGridlet()方法，创建一个或多个网格任务
  2. 调用createGridUser()方法，创建一个或多个网格用户
  3. 调用printGridletList()方法，打印任务列表

}

/\*\*使用两种方式创建网格任务，手动创建或借助GridSimRandom类\*/

**private static** GridletList createGridlet(){

1. 创建一个网格任务集合，new GridletList()；
2. 手动创建3个（数量自己定就好，例子中创建了3个）任务，4个参数手动赋值，并添加至任务集合；
3. 借助GridSimRandom和GridSimStandardPE类循环创建5个任务，id递增，其他3个参数在一定范围内随机生成，任务创建完毕添加至集合；
4. 返回网格任务列表

}

/\*\*创建网格用户，并将其指派给网格任务，这种创建用户方式好像不常用\*/

**private static** ResourceUserList createGridUser(GridletList list){

1. 创建一个资源用户集合，new ResourceUserList()；
2. 向用户集合添加新用户，userList.add(int userID)；
3. 将用户ID分配给指定的任务（一个for循环，gl.setUserID(int id)）；
4. 返回用户集合

}

/\*\*打印任务列表，包含：任务ID，用户ID，任务长度，文件大小和输出大小\*/

**private static void** printGridletList(GridletList list){

打印第一行的属性，然后for循环，打印每个任务的各个属性值；

}

}

**Example3**

功能：两个GridSim实体之间互相联系，从一个实体发送网格任务到另一个实体，并更改任务状态值

说明：在本例中，创建了两个类，一个是Example3，一个是Test。两个类都继承了GridSim

程序结构：

**class** Test **extends** GridSim{

Test(String name, **double** baud\_rate) **throws** Exception{

分配一个新的Test对象，调用父类的构造方法，super（name，baudRate）；

}

**public void** body(){

for循环，一次处理一个事件{

1. 从事件中获得data域（即网格任务）；
2. 更改网格任务的状态码为Gridlet.SUCCESS；
3. 将网格任务发回给源实体；

}

调用父类方法，终止输入输出实体；

}

}

**class** Example3 **extends** GridSim{

Example3(String name, **double** baud\_rate, GridletList list) **throws** Exception{

1. 分配一个新的Example3实体，调用父类构造方法super（name）；
2. 给成员变量赋值；
3. 创建一个Test实体；

}

**public void** body(){

for循环，每次处理一个网格任务，随事件发送给Test实体{

1. 从集合中取出一个网格任务；
2. 调用super.send()方法，向网格用户Test发送网格任务；
3. 接收发回来的网格任务（此时任务状态已改为SUCCESS）；
4. 将任务添加到收到的任务列表；

}

循环结束后向对方发送仿真结束的信息；

}

**public** GridletList getGridletList(){

返回一个网格任务集合（从Test实体收到的网格任务）；

}

**public static void** main(String[] args){

1. 初始化GridSim包；
2. 创建网格任务列表；
3. 创建Example3对象（同时也会创建Test对象）；
4. 开始仿真；
5. 获取新的网格任务列表（即从Test返回的任务列表），并打印；

}

/\*\*创建8个网格任务，并储存至集合返回\*/

**private static** GridletList createGridlet(){

1. 创建容器；
2. 手动创建3个网格任务，并添加至集合；
3. 随机创建5个网格任务，并添加至集合；
4. 返回网格任务列表；

}

**private static void** printGridletList(GridletList list){

循环打印每个网格任务的ID和状态，以便查看是否每个任务都是SUCCESS的；

}

}

**Example4**

功能：一个用户创建8个网格任务，并将任务发送至指定资源名的资源实体（资源ID由SimJava自动分配），资源实体将更改任务状态为Gridlet.SUCCESS

说明：程序大致的执行过程是：main方法开始，初始化GridSim包；创建一个网格资源对象；然后创建一个Example4对象（就是一个网格用户对象），同时创建网格任务并添加到任务列表；然后开始仿真，执行body方法；主线程用一个while(true)循环等待网格资源完成向GIS的注册，并获取到资源列表；向资源发送send请求获取资源属性；for循环，遍历任务列表中的任务，并将其一一提交至网格资源；接收从资源返回的，更改了任务状态的网格任务，并将其添加至receiveList\_列表；仿真结束，回到main方法打印接收到的网格任务。

程序结构：

**class** Example4 **extends** GridSim{

Example4(String name, **double** baud\_rate) **throws** Exception{

1. 给成员变量赋值；
2. 为GridSim实体指定ID（用户ID）；
3. 调用createGridlet（）方法，创建一个网格任务列表；

}

**public void** body(){

一个while(true)循环，等待得到资源列表（多线程，实体需先向GIS注册，才

能被获取）{

1. 先暂停1秒，等待资源实体完成注册；
2. 获取资源列表；
3. if判断，如果资源数>0{
4. 向资源发请求，希望获取其属性（RESOURCE\_CHARACTERISTICS）；
5. 获取资源属性，并将该事件记录到txt文件；
6. 跳出while循环（以上代码完成，说明已经成功获取资源列表）；

}else{

打印文字，继续等待获取资源列表；

}

}

for循环{

1. 从网格任务列表获取一个网格任务；
2. 将任务发送至具有指定ID的资源实体；
3. 将事件记录到txt文件；
4. 接收从资源实体返回的网格任务；
5. 记录事件；
6. 将收到的任务添加到receiveList\_集合；

}

各种关闭；

}

**public** GridletList getGridletList(){

返回接收到的任务列表；

}

**private** GridletList createGridlet(**int** userID){

1. 手动创建3个任务，并添加到任务集合；
2. 随机创建5个任务，并添加到任务集合；
3. 返回任务集合；

}

**public static void** main(String[] args){

1. 初始化GridSim包（参数用户总数，一定一定不要不顾实际情况乱填值！）；
2. 创建1个资源对象；
3. 创建Example4对象；
4. 开始仿真；
5. 仿真结束，打印网格任务；

}

**private static** GridResource createGridResource(String name){

1. 创建机器列表，存储一个或多个机器（MachineList，继承LinkedList<Machine>）；
2. 创建一个机器，并将其添加至机器列表（Machine(3个参数)）；
3. 若需要创建更多机器，重复第二步（代码一样！）；
4. 创建一个资源属性对象（ResourceCharacteristics），存储网格资源特性；
5. 创建一个网格资源对象（需要捕获异常，防止未初始化GridSim包，或实体名为空，或PE数为0）；
6. 返回一个网格资源；

}

**private static** void printGridletList(GridletList list){

循环打印每个网格任务的ID、状态、资源ID和开销；

}

}

**Example5**

功能：一个网格用户创建8个网格任务，并将任务随机发送给多个网格资源中的一个（如果将分配策略由随机改为其他，则可以实现用不同的任务调度算法分配任务）

说明：搞清楚了一件事，接收到的网格资源集合里面存储的是Integer对象，这是包装过的网格资源ID，所以资源集合中存储的不是资源实体。想要通过资源ID获取资源名、资源开销等参数，必须向资源发送send请求，获取指定ID的资源的资源属性对象，然后由资源属性对象获得资源的其他属性。

另外，若想使用不同的任务调度策略，只需要修改body中的相关内容。

程序结构：

**class** Example5 **extends** GridSim{

Example5(String name, **double** baud\_rate, **int** total\_resource)**throws** Exception{

1. 这里参数比之前的案例构造方法多了一个，多的参数代表资源总数；
2. 给成员变量赋值（成员变量也比以前案例多一个totalResource\_）；
3. 为GridSim实体指定ID（用户ID）；
4. 调用createGridlet（）方法，创建一个网格任务列表；

}

**public void** body(){

一个while(true)循环获取资源列表{

此时判断是否跳出循环的条件是获取到的资源总数是否与成员变量totalResource相等，若相等则跳出while循环，若不等，则输出“等待获取资源列表”的消息；

}

一个for循环，遍历资源列表中的资源{

1. 从资源列表中获取第i个资源，并得到它的ID值，存入一个数组中；
2. 发送send请求，获取资源的属性（因为这里用到资源ID来区分资源，因此上一句是有必要的）；
3. 由资源属性得到资源名和资源开销（因为资源列表中存储的是资源ID而不是资源实体，因此想要获得资源的除ID的其他属性，必须先得到资源属性对象！！！），并存入对应数组；

}

一个for循环，遍历任务列表中的任务（在这里使用不同的调度算法）{

1. 从列表中取出第i个任务；
2. 用Random对象的nextInt方法，随机生成一个[0, totalResource)的整数，作为资源下标（资源id是SimJava分配的，不是连续的，因此不能直接将其作为资源id）；
3. 将任务提交给数组中对应下标的资源实体（gridletSubmit方法）；
4. 接收从资源返回的任务，并将其添加到receiveList\_中；

}

仿真结束，各种关闭；

}

/\*\*用户向资源提交任务之后，资源会更改任务状态，并将任务回传给用户\*/

**public** GridletList getGridletList(){

返回接收到的任务列表（将在仿真结束之后，在main方法中调用）；

}

**private** GridletList createGridlet(**int** userID){

1. 手动创建3个任务，并添加到任务集合；
2. 随机创建5个任务，并添加到任务集合；
3. 返回任务集合；

}

**public static void** main(String[] args){

1. 初始化GridSim包；
2. 创建3个资源对象（调用3次createGridResource方法，这要看仿真要求）；
3. 创建1个Example5对象（只有一个用户）；
4. 开始仿真；
5. 仿真结束，打印网格任务；

}

**private static** GridResource createGridResource(String name){

1. 创建机器列表，存储一个或多个机器（MachineList，继承LinkedList<Machine>）；
2. 创建一个机器，并将其添加至机器列表（Machine(3个参数)）；
3. 若需要创建更多机器，重复第二步（代码一样！）；
4. 创建一个资源属性对象（ResourceCharacteristics），存储网格资源特性；
5. 创建一个网格资源对象（需要捕获异常，防止未初始化GridSim包，或实体名为空，或PE数为0）；
6. 返回一个网格资源；

}

**private static void** printGridletList(GridletList list){

循环打印每个网格任务的ID、状态、资源ID和开销；

}

}

**Example6**

功能：创建3个网格资源和3个网格用户，每个用户有3~7随机个网格任务。用户将任务随机提交给资源，资源更改任务状态，并将其返回给用户

说明：本例创建3个用户，每创建一个用户，在仿真开始的时候都会执行body方法，因此本例的body方法执行了3次（并且应该是并行执行的）。

本例中所有用到随机值的地方全部使用GridSimRandom类中的方法，与以往案例不太一样。以往案例都要先创建Random对象，然后调用对象的方法生成int或double；本例中使用的都是GridSimRandom的静态方法（详细可参看Example6的body()和createGridlet()方法）

程序结构：

**class** Example6 **extends** GridSim{

Example6(String name, **double** baud\_rate, **int** total\_resource) **throws** Exception{

1. 最后一个参数代表资源总数（若只需要创建一个资源就不用设该参数了）；
2. 给成员变量赋值；
3. 为GridSim实体指定ID（用户ID）；
4. 调用createGridlet（）方法，创建一个网格任务列表；

}

/\*\*每创建一个网格用户，就会调用一次body方法，因此本例会调用3次\*/

**public void** body(){

一个while(true)循环获取资源列表{

此时判断是否跳出循环的条件是获取到的资源总数是否与成员变量totalResource相等，若相等则跳出while循环，若不等，则输出“等待获取资源列表”的消息；

}

一个for循环，遍历资源列表中的资源{

1. 从资源列表中获取第i个资源，并得到它的ID值，存入一个数组中；
2. 发送send请求，获取资源的属性（因为这里用到资源ID来区分资源，因此上一句是有必要的）；
3. 由资源属性得到资源名和资源开销（因为资源列表中存储的是资源ID而不是资源实体，因此想要获得资源的除ID的其他属性，必须先得到资源属性对象！！！），并存入对应数组；

}

一个for循环，遍历任务列表中的任务（在这里可以使用不同的调度算法）{

1. 从列表中取出第i个任务；
2. 调用GridSimRandom的静态方法intSample方法，随机生成一个[0, totalResource)的整数，作为资源下标（资源id是SimJava分配的，不是连续的，因此不能直接将其作为资源id）；
3. 将任务提交给数组中对应下标的资源实体（gridletSubmit方法）；
4. 接收从资源返回的任务，并将其添加到receiveList\_中；

}

仿真结束，各种关闭；

}

**public** GridletList getGridletList(){

返回接收到的任务列表（将在仿真结束之后，在main方法中调用）；

}

/\*\*该方法与以往例子中生成随机任务的方法稍有不同，见本例的说明\*/

**private** GridletList createGridlet(**int** userID){

* + - 1. 手动创建3个网格任务，分别绑定用户ID，添加到任务集合；
      2. 调用GridSimRandom.intSample(5)方法，产生一个0到4的整数，作为随机产生的任务总数count，因此总任务数为count+3；
      3. for循环，生成count个任务（任务的长度，文件大小，输出大小的值都是随机的）；
      4. 返回任务集合；

}

**public static void** main(String[] args){

1. 初始化GridSim包（注意，初始化时需指定网格用户数，以往不创建用户对象或者只有一个用户，而本例需要创建3个网格用户对象）；
2. 创建3个资源对象（调用3次createGridResource方法，这要看仿真要求）；
3. 创建3个Example6对象（一共3个用户）；
4. 开始仿真；
5. 仿真结束，打印每个用户的网格任务；

}

**private static** GridResource createGridResource(String name){

1. 创建机器列表，存储一个或多个机器（MachineList，继承LinkedList<Machine>）；
2. 创建一个机器，并将其添加至机器列表（Machine(3个参数)）；
3. 若需要创建更多机器，重复第二步（代码一样！）；
4. 创建一个资源属性对象（ResourceCharacteristics），存储网格资源特性；
5. 创建一个网格资源对象（需要捕获异常，防止未初始化GridSim包，或实体名为空，或PE数为0）；
6. 返回一个网格资源；

}

**private static void** printGridletList(GridletList list, String name){

循环打印某用户（参数指定的）的每个网格任务的ID、状态、资源ID和开销；

}

}