# Simple to Program

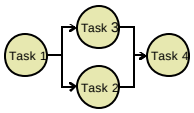
```
public class COG implements StatusListener{
  public void create() { … }
  public void submit () { … }
  public void statusChanged (StatusEvent e)  { … }
  public static void main (String arg[]){
                                  try {
        COG cog = new COG();
        cog.create();
        cog.submit();
      catch (Exception e) {
        logger.error("Something went wrong:", e);
  }
}
```

```
public void statusChanged (StatusEvent event) {

  Status status  = event.getStatus( );

  logger.debug(``Status changed to " +  status.getStatusCode());

  if (status.getStatusCode ( ) == Status.COMPLETED) {
    logger.info(``Task Done");
  elsif (status.getStatusCode ( ) == Status.FAILED) {
    logger.info(``Task Failed");
    System.exit(1);
  }
}
```




```
TaskGraph tg = new TaskGraphImpl();

public void create () {
 // define tasks
  …..
 /* Add the tasks to the TaskGraph */
 tg.add(task1);
 tg.add(task2);
 tg.add(task3);
 tg.add(task4);
 tg.addDependency(task1, task2);
 tg.addDependency(task1, task3);
 tg.addDependency(task2, task4);
 tg.addDependency(task3, task4);
}

public void submit() {
 TaskGraphHandler handler =  new
                        TaskGraphHandlerImpl ();
    try {
      handler.submit(tg);
    } catch (Exception e) {
      logger.error(``Some Error occured", e);
      System.exit(1);
    }
}
```

```
try {
    handler.submit (cog);
  } catch (InvalidSecurityContextException  ise) {
    logger.error(``Security Exception", ise);
    System.exit(1);
  } catch (TaskSubmissionException tse) {
    logger.error(``TaskSubmission Exception", tse);
    System.exit(1);
  } catch (IllegalSpecException ispe) {
    logger.error(``Specification Exception", ispe);
    System.exit(1);
  } catch (InvalidServiceContactExceptionisce){
    logger.error(``Service Contact Exception", isce);
    System.exit(1);
  }
```

## CoG Kit Patterns

The philosophy behind the Java CoG Kit is based on object-oriented programming patterns. One of these patterns is the "task" pattern to help conducting a workflow in the Grid. An implementation can be achieved by reusing the interfaces and implementations provided by the Java CoG Kit.

## A simple COG

We show an example of a class named COG that is based on our task pattern. The task pattern always contains a method to create and submit tasks, and a method that reports on status changes.

## Completing the COG

The Java CoG Kit contains interfaces and methods that simplify the definition and use of task graphs and tasks based on our abstraction model. Defining a handler that deals with the submission of the job becomes as simple as our example demonstrates. A comparable C program would consume many more lines.

More sophisticated handlers can be custom designed.

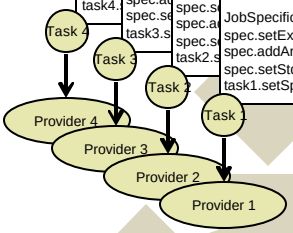
Tasks are specified through a simple task specification.

## Providers & Binding

Providers for various Grid toolkits can be seamlessly integrated and a task can be bound to a particular Grid resource. Through this process late binding is possible.

```
Task task4 = new Task();
Task task3 = new Task();
Task task2 = new Task();
Task task1 = new Task();

JobSpecification spec = new JobSpecificationImpl();
spec.setExecutable("/share/bin/climate");
spec.addArguments("-dimension 512 512 512 –task 1");
spec.setStdOutput("output1.txt");
task1.setSpecification(spec);
```




## Even simpler

The same example can be formulated even simpler through our Java CoG Kit workflow language. Karajan, our workflow engine, will take care of the rest.

```
Service service = new ServiceImpl(Service.JOB_SUBMISSION);
Service service = new ServiceImpl(Service.JOB_SUBMISSION);
Service service = new ServiceImpl(Service.JOB_SUBMISSION);
Service service = new ServiceImpl(Service.JOB_SUBMISSION);

service.setProvider(``GT3_2_1");

// Set Security Context – e.g. certificates and such
SecurityContext  securityContext  = CoreFactory.newSecurityContext(``GT3_2_1");
securityContext.setCredentials(null); // e.g. set it to default in ./globus
service.setSecurityContext(securityContext);
// Set Contact – e.g. where to go to
ServiceContact serviceContact =
    new ServiceContactImpl ( http://hot.mcs.anl.gov:8080/ogsa/services/base/gram/,
    MasterForkManagedJobFactoryService");
service.setServiceContact(serviceContact);

task1.setService(Service.JOB_SUBMISSION_SERVICE, service);
```

```
<project>
    <include file="sysdefaults.xml"/>
    <sequential>
        <gridExecute executable="/share/bin/climate"
            arguments="-dimension 512 512 512 -task 1"
            host="hot.mcs.anl.gov'
            provider="gt3.2.1"
            stdout="output1.txt"/>
        <parallel>
            <gridExecute executable="/share/bin/climate"
                arguments="-dimension 512 512 512 -task 2"
                host="cold.mcs.anl.gov'
                provider="gt2"
                stdout="output2.txt"/>
            <gridExecute executable="/share/bin/climate"
                arguments="-dimension 512 512 512 -task 3"
                host="hot.mcs.anl.gov'
                provider="gt3.2.1"
                stdout="output3.txt"/>
        </parallel>
        <gridExecute executable="/share/bin/climate"
            arguments="-dimension 512 512 512 -task 4"
            host="blue.mcs.anl.gov'
            provider="gt4"
            stdout="output4.txt"/>
    </sequential>
</project>
```