Class not found in distributed task

1. Issue description

Dist task has dependency to classes which are available on client side which executes task on remote space.
Client is deployed as stateless PU. Dependent classes are not loaded eagerly by space PU because they are accessed in executed method only when some condition is met (normal case when you have some business logic).

2. Dist task

```
public class DistTaskNotLoadingAllClasses implements DistributedTask<String, String> {
    private static final Logger logger =
Logger.getLogger(DistTaskNotLoadingAllClasses.class.getName());
    private final int branch;

    public DistTaskNotLoadingAllClasses(int branch) {
        this.branch = branch;
    }

    @Override
    public String execute() throws Exception {
        switch (branch) {
            case 1:
                logger.info("creating instance of " + new Class1());
                return "branch" + branch;
            case 2:
                logger.info("creating instance of " + new Class2());
                return "branch " + branch;
            default:
                throw new RuntimeException("unsupported branch=$branch");
        }
    }
 // reduce
}
```

3. How to reproduce

1. space is deployed as separate pu
2. deploy pu
3. pu executes dist task with branch 1, Class1 is loaded
4. pu is undeployed
5. pu is deployed on on same or different gsc (in both cases we will hit issue in next step)
6. pu executes dist task with branch 2, Class2 is not found

Example log from pu gsc

```
2016-09-21 09:47:01,962 GSC SEVERE [com.gigaspaces.grid.gsc] - Failed to instantiate sample-pu-1
[1]; Caused by: org.jini.rio.core.JSBInstantiationException: java.lang.ClassNotFoundException:
DefaultClassProvider [7835556513865216316] could not locate required class
[org.asl.gigaspaces.Class2] at the specified class loader [3]
        at
com.gigaspaces.lrmi.classloading.DefaultClassProvider.getClassDefinition(DefaultClassProvider.java:10
at
com.gigaspaces.lrmi.classloading.IClassProviderGigaspacesMethodinternalInvoke2.internalInvoke(Unknown
Source)
        at com.gigaspaces.internal.reflection.fast.AbstractMethod.invoke(AbstractMethod.java:41)
        at com.gigaspaces.lrmi.LRMIRuntime.invoked(LRMIRuntime.java:477)
        at com.gigaspaces.lrmi.nio.Pivot.consumeAndHandleRequest(Pivot.java:573)
        at com.gigaspaces.lrmi.nio.Pivot.handleRequest(Pivot.java:667)
        at com.gigaspaces.lrmi.nio.Pivot$ChannelEntryTask.run(Pivot.java:196)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1145)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:615)
        at java.lang.Thread.run(Thread.java:744)
```

4. Solutions/Ideas worth consider

1. Whenever client execute dist task to be executed, and any class is not loaded, GS tries to load classes from class loaded associated with client who initiate this execution.
2. Whenever PU is undeployed then class loaded associated with this PU on space side should not be consider for any loading of missing classes.

5. How we workaround this issue currently

We tries to guarantee that whenever we sent dist task to be executed we load all classes eagerly.
For that we tries to access all dependent classes from static block in dist task class, e.g.

```
class DistTask {
  static {
    eagerlyLoad(Class1.class, Class2.class);
  }
}
// implementation of eagerlyLoad
public static void eagerlyLoad(Class<?>... classes) {
  for (Class<?> aClass : classes) {
    Class.forName(aClass.getName());
  }
}
```

Limitation of this solution is that we cannot use anonymous classes since they have names generated (and we do not want to have mess in our code like `Class.forName("com.example.ComeClass$1")` etc.)
This will be even bigger problem when we start using Java 8 and lambdas.
Whenever we face this issue because we forgot to add class to static block we have to restart all spaces which holds RMI Class Loaders.