

Slightly More Interesting Target

```
// Add the following include files
#include "InfString.h"
#include "GalIter.h"

// Modify the setup method
void MyTarget::setup() {

    // An infinite string class
    InfString list;

    // Construct a list of the blocks at the top level
    // using an iterator.
    GalTopBlockIter blockIterator(*galaxy());
    Block *b;
    while ((b = blockIterator++) != 0) {
        list << b->name() << "\n";
    }

    // Display the names
    Error::message(list);
}
```

The World's Simplest Target — Compilation

The following extremely simple makefile will work

```
MyTarget.o:MyTarget.cc
```

```
    g++ -I$(PTOLEMY)/src/kernel -c MyTarget.cc
```

The World's Simplest Target — Instantiation

```
// A class that keeps track of all available Targets:
#include "KnownTarget.h"

// Create an instance of the Target (there has to be
// at least one instance from which clones are made).
// The arguments are:
//     Target name
//     types of stars supported
//     a short description of the Target
static MyTarget myTargetPrototype("MyTarget", "SDFStar",
    "The world's simplest Target");

// Register the Target in the list of known Targets.

static KnownTarget entry(myTargetPrototype,"MyTarget");
```

The World's Simplest Target — Method Definitions

```
// These methods just use the Error class to issue messages

void MyTarget::setup() {
    Error::message("Requesting setup\n");
}

int MyTarget::run() {
    Error::message("Requesting run\n");
    return 1;
}

void MyTarget::wrapup() {
    Error::message("Requesting wrapup\n");
}

// The method to return a new instance
Block* MyTarget::makeNew() const {
    return new MyTarget(name(), starType(), descriptor());
}
```

The World's Simplest Target — Class Definition

```
// The base class template file must be included
#include "Target.h"

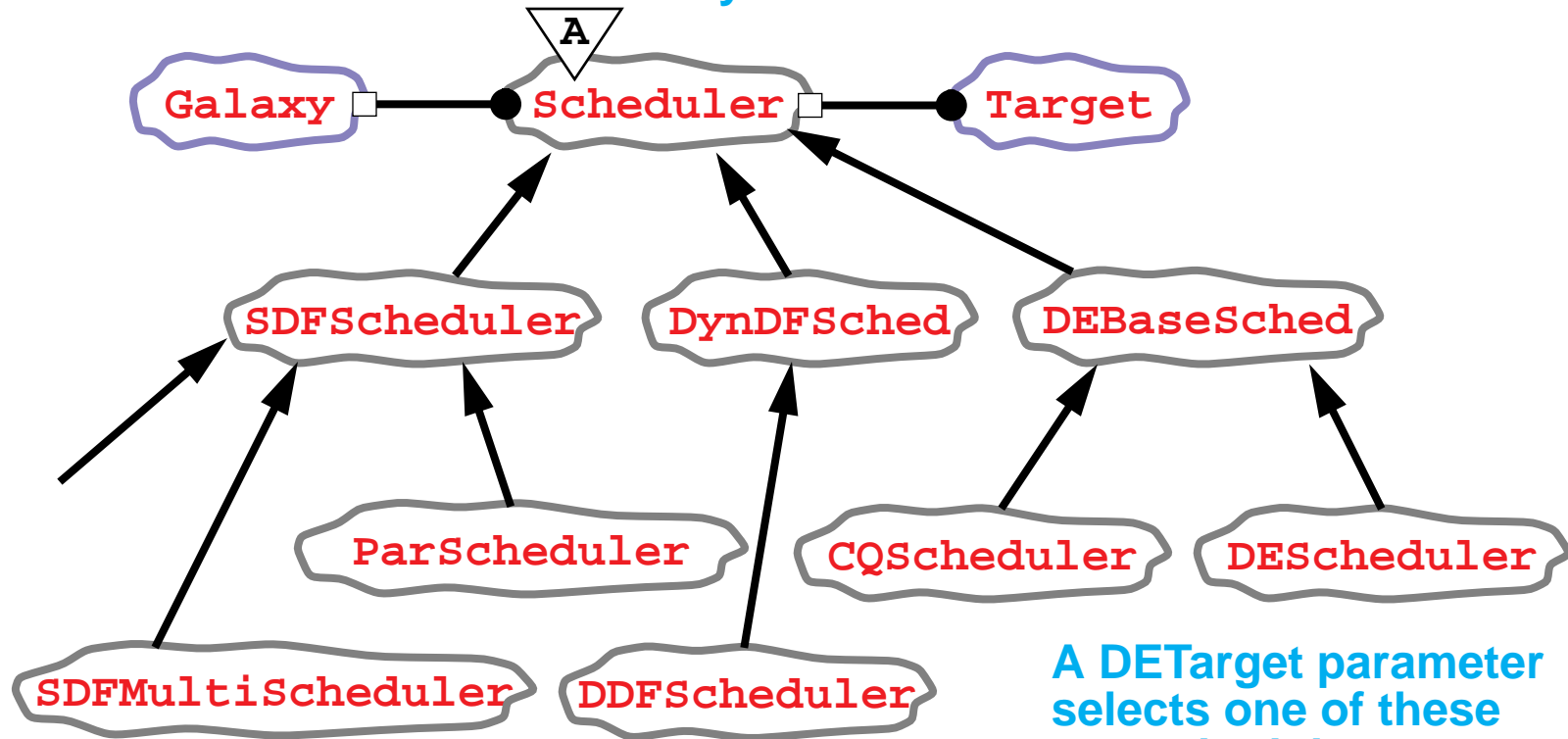
class MyTarget : public Target {
public:
    void setup();
    int run();
    void wrapup ();

    // The constructor simply passes the arguments on
    // to the base class constructor.
    MyTarget(const char* nam,const char* stype
             const char* desc) : Target(nam,stype,desc) {}

    // Every Target has a method to make another instance
    // of the same type of Target.
    Block* makeNew() const;
};
```

Scheduler Class Hierarchy

A Scheduler contains a Galaxy



Scheduler methods

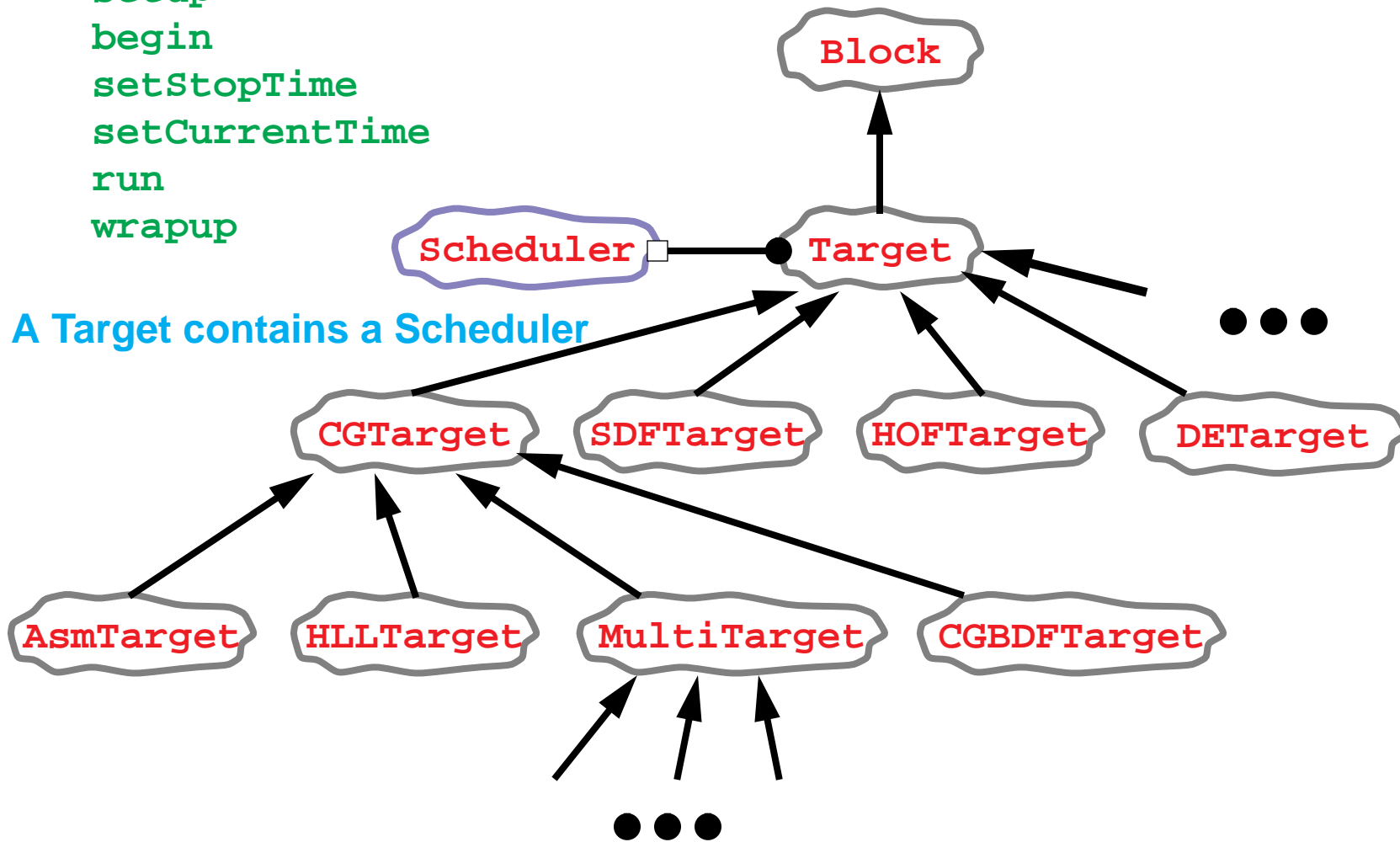
```
setup
setStopTime
setCurrentTime
run
```

A DETarget parameter selects one of these two schedulers

Target Class Structure

Target methods

setup
begin
setStopTime
setCurrentTime
run
wrapup



Defining Targets

- **A Target supervises the execution of Universe**
- **A Universe has a Target that manages its execution**
- **A Target normally contains a Scheduler**
- **The Target is a Block, and hence can have States**
- **A Target can do simulation, code generation, or compilation**
- **A Target can have child Targets (e.g. for multiprocessors)**