Conversation with the customer:

Client clarification:

The production code must implement, at a minimum, the classes/methods listed in the specifications below. Please ensure you use the given class names, method names, and parameter names. You are free to implement additional classes and methods.

Client Clarification 2:

Values raised by exceptions are explanatory strings having the following format:

name of the class raising the exception	followed by
a period	followed by
name of the method raising the exception	followed by
a colon and two spaces	followed by
a useful diagnostic message of your choice	

Client Clarification 3:

Method interfaces should be constructed so that parameters can be specified by position (where possible) or name.

Specifications

Background: We are continuing to add components to the satellite simulator with this assignment. Specifically, we are adding a component which controls the various parts of the satellite, a component that logs information, and a "place holder" component that generates fake data.

Functional Requirements:

class Controller

Abstraction: *Controller* represents a supervisory component that oversees the operation of all the devices on the satellite. Our satellite architecture is designed such that devices -- star sensor, for example -- respond to requests for information. They cannot provide information unless requested to do so. The *Controller* is the "brain" of the satellite. It communicates with each device in a prescribed order within a prescribed timeframe.

constructor	description
CA03-1.1 __init__	Controller()

Creates an instance of *Controller*.

Parameters: No parameters

Returns: An instance of *Controller*.

State change: The instance contains no specfic information

Exceptions: No exceptions

instance methods	description
CA03-1.2 initialize	initialize(architectureFile)

Informs the controller about the satellite architecture.

Parameters: "architectureFile" is the name of an XML file from which to load information about satellite devices the controller needs to manage. It is a string of the form f.xml, where "f" is a file name having a length .GE.1 and ".xml" is the literal file extension. Mandatory. Arrives unvalidated.

Returns: A list of unique devices in the architecture file (as indicated by the "frame" XML tag). Returns [] the frame is empty.

State change: The controller instantiates components described in the architecture file.

Exceptions:

Type:	ValueError
Rasied when:	The file is invalid. Examples: - the file name violates the parameter specifications - no file exists by the specified file name - the file contains invalid information
Exit conditions:	The controller remains in the state it is was in before the method was invoked.

Implementation Note: The architecture file is of the form:

```
<configuration>

<definition component="Environment">
<parm name="rotationalPeriod">10000000</parm>
</definition>

<definition component="Monitor">
<parm name="logFile">logfile.txt</parm>
</definition>

<definition component="StarSensor">
<parm name="fieldOfView">0.174533</parm>
<parm name="starFile">SAOChart.txt</parm>
</definition>

<definition component ="Device">
</definition>

<frame rate="1000000">
<device>Device</device>
<device>StarSensor</device>
```

<device>Device</device>
</frame>

</configuration>

The meaning of the XML tags are:
- "configuration" designates the start and top of the simulation settings. It only occurs once. No other tags appear outside the <configuration> and its corresponding end tag, </configuration>.
- "definition" designates a software component that should be instantiated. The "component" attribute gives the name of the component.
- "parm" is an optional tag that occurs within the "definition" tag. Each "parm" tag has an attribute that corresponds to a parameter name. The body of the "parm" tag gives the value to be passed as the parameter. Multiple "parm" tags may be present.
- "frame" describes the satellite devices that will be polled for information (see the run() method below for more information). This tag can only occur once. An exception should be raised if multiple frames are defined.
- "device" is the type of component that is to be polled within a frame.

Note: The "definition" and "frame" tags can be in any order.

Note: An exception should be raised if the information in the XML is invalid or incomplete. For example, each of the following is invalid:

<definition component="Siren">
</definition>
why: No component called "Siren" exists

<definition component="StarSensor">
</definition>
why: StarSensor cannot operate without a field of View and a starCatalog

<definition component="Monitor">
<parm name="file">logfile.txt</parm>
</definition>
why: Monitor does not have a parameter named "file"

<frame rate="abcef">
</frame>
why: The "rate" attribute has an invalid value.

Note: An exception should be raised if an device in the frame does not have a corresponding definition.

~~CA03-1.3~~ ~~configure~~ ~~configure(environment)~~

~~Passes information (such as the simulated time, etc.) about the simulation environment to the controller.~~

~~Parameters:~~	~~"environment" is an instance of *Environment* that contains information about the system being simulated. Mandatory. Arrives unvalidated.~~
~~Returns:~~	~~*True*~~
~~State change:~~	~~The controller retains the environment object for later reference.~~
~~Exceptions:~~	~~Type:~~ ~~ValueError~~
	~~Rasied when:~~ ~~The interface contract above is violated.~~
	~~Exit conditions:~~ ~~The instance remains in the state it is was in before the method was invoked.~~

CA03-1.4 run run(microseconds)

Runs the simulation for the specified number of microseconds.

Parameters:	"microseconds" is the amount of simulated time the controller is to run. Integer .GT. 0. Mandatory, Arrives unvalided.
Returns:	The number of microseconds (an integer) that were simulated by the run method.
State change:	The controller retains all necessary information to continue normal operation if restarted.
Exceptions:	Type: ValueError
	Rasied when: The interface contract above is violated or if the controller has not been properly initialized with an architecture file that specifies the monitor and environment.
	Exit conditions: The instance remains in the state it is was in before the method was invoked.

Implementation Note: The controller repeatedly calls the serviceRequest() of connected devices at specified intervals, referred to as "frames." For example, suppose we have a satellite whose frame structure is described by the following XMLe:

<frame rate="1000000">
<device>Device</device>
<device>StarSensor</device>
<device>Device</device>

</frame>

The "rate" attribute states that the devices in the frame are polled every 1000000 microseconds. Since the simulated clock beings at time 0, the controller calls Device.serviceRequest() at time 0. Device advances the clock to simulate its response time and returns a value to the controller at time 40. The controller calls StarSensor.serviceRequest() at time 40. It returns a value to the controller at time 80. The controller calls Device.serviceRequest() at time 80 and receives a return value at time 120. Once we've contacted all the devices in the frame, we check to see if the amount of time specified by rate has expired since the last time we started a frame. If so, we repeat calling the serviceRequest() method of each device. If not, we wait until we are eligible to start the frame -- at time 1000000, in this case, -- and then poll each device. This continues until both the end of the frame and the designated stop time have been been reached.

Note that the value of the "device" tag names the class of the device whose serviceRequest() method is called. There is only one instance for each unique value associted with "device". In the example above, the serviceRequest() method of the instance of Device is called, then the serviceRequest() method of the instance of StarSensor is called, then the serviceRequest() method of the instance of Device is called again.

The run() method polls all devices in the frame before checking if the simulation duration has been reached. In other words, the simulation does not stop part way through a frame if the specified amount of time to run the simulation has been reached. If *run(microseconds=40)* were used with the frame defined above, the simulation would stop at time 120.

The "rate" attribute of frame describes the minimum amount of time that must expire from the start of one frame to the next. The clock time of each frame begins on a multiple of the frame rate if the time required to service requests of all the devices in the frame is less than the frame rate (as in the example above). If the time to service requests for all devices exceeds the frame interval, the next frame begins as soon as the last device is polled. In other words, the polling is not interrupted midframe if the frame interval has been exceeded. If <frame rate="10"> were used in the example above, the first frame would begin at time 0, the next frame would begin at time 120, the next frame would begin at time 240, etc.

class Monitor

Abstraction: *Monitor* is a device whose purpose is to observe and record the following satellite events: 1) requests the controller makes for service, and 2) data returned to the controller in response to a service request

	constructor	description
CA03-2.1	__init__	Monitor()

Creates an instance of the device.

Parameters:	No parameters
Returns:	An instance of *Monitor*
State change:	The instance contains no specfic information
Exceptions:	No exceptions

	instance methods	description
CA03-2.2	initialize	initialize(logFile)

Informs the monitor where to store log information.

Parameters:	"logFile" is the name of a text file to which log information is recorded. It is a string of the form f.txt, where "f" is a file name having a length .GE.1 and ".txt" is the literal file extension. Mandatory. Arrives unvalidated.	
Returns:	True if "logFile" is a new file; False if "logFile" exists.	
State change:	A new file is used to log information if no file exists with the specified file name; otherwise, log information is appended to the existing file.	
Exceptions:	Type:	ValueError
	Rasied when:	The file name violates the parameter specfications.
	Exit conditions:	The instance remains in the state it is was in before the method was invoked.

CA02-2.3	configure	configure(environment)

Passes information (such as the simulated time, etc.) about the simulation environment to the device.

Parameters:	"environment" is an instance of *Environment* that contains information about the system being simulated. Mandatory. Arrives unvalidated.	
Returns:	*True*	
State change:	The device retains the environment object for later reference.	
Exceptions:	Type:	ValueError
	Rasied when:	The interface contract above is violated.
	Exit conditions:	The instance remains in the state it is was in before the method was invoked.

CA03-2.4	serviceRequest	serviceRequest(source, target, event) Logs simulated time, the device which originated the data, and the data itself. Parameters: "source" is the name of the device that originated the event. String. Mandatory. Arrives Validated. "target" is the name of the device to which the event is intended. String. Mandatory. Arrives Validated. "event" is the information associated with the event. Optional, defaults to "serviceRequest" if missing. Arrives Validated. Returns: The integer time of the simulated clock. State change: The information is recorded to the log file. The information is discarded if no log file has been previously designated. Exceptions: Type: ValueError Rasied when: The interface contract above is violated. or if the monitor has not been properly initialized with a log file or environment. Exit conditions: The instance remains in the state it is was in before the method was invoked. File details: Each call to serviceRequest() results in a single line being written to the log file. The format of each line is: time followed by tab followed by source followed by tab followed by target followed by tab followed by event Example, given the architecture file described above 0\tController\tDevice\tserviceRequest 40\tDevice\Controller\t0000 40\tController\tStarSensor\tserviceRequest 80\tStarSensor\tController\t1234 80\tController\tDevice\tserviceRequest 120\tDevice\tController\t1111 1000000\tController\tDevice\tserviceRequest 1000040\tDevice\tController\t0000 etc
	class Device	
	Abstraction:	*Device* is a generic satellite component that produces simulated results.
	constructor	description
CA03-3.1	__init__	Device() Creates an instance of the device. Parameters: No parameters Returns: An instance of *Device* State change: The instance contains no specfic information Exceptions: No exceptions
	instance methods	description
CA03-3.2	configure	configure(environment) Passes information (such as the simulated time, etc.) about the simulation environment to the device. Parameters: "environment" is an instance of *Environment* that contains information about the system being simulated. Mandatory. Arrives unvalidated. Returns: *True* State change: The device retains the environment object for later reference. Exceptions: Type: ValueError Rasied when: The interface contract above is violated. Exit conditions: The instance remains in the state it is was in before the method was invoked.
CA03-3.3	serviceRequest	serviceRequest() Returns simulated bogus data. Parameters: No parameter Returns: A four-character lowercase string representing a hexadecmial value ranging from "8000" (i.e., -32767 decimal in one's complement) to "7fff" (i.e., 32767 decimal). The value should have the following property: a 25% chance of being "0000" a 50% chance of being a uniformly-distributed random value between "0001" and "7fff" a 25% chance of being a uniformly-distributed random value between "fffe" and "8000" State change: This operation takes 40 microseconds of simulated time to complete, meaning, the simulated clock is incremented by 40 microseconds before serviceRequest() returns a result. Exceptions: Type: ValueError

		Rasied when:	configure() has not yet been called.
		Exit conditions:	The instance remains in the state it is was in before the method was invoked.