Jacob Baird
3555993
CS4015
Final Project

Seismographic Monitoring Software

Seismographic Monitoring Software (SMS) provides users an interface to store, retrieve and analyze real-time seismographic data from distributed seismometers, as well as to generate earthquake warnings.

SMS implements the following architectural patterns:

Model-View-Control Pattern:

This pattern is motivates by the requirement that the user may interact with seismographic data via a user interface.

Event-Bus Pattern:

This pattern is motivated by the requirement that the user may generate earthquake warnings and send these warnings to an earthquake warning system.

Pipe-Filter Pattern:

This pattern is motivated by the requirement that distributed seismometers generate real-time seismographic data and this data is sent to the user.

SMS implements the following design patterns:

Abstract Factory

SMS implements the abstract factory pattern through the warning station generating tsunami or earthquake alerts. Alerts may be warnings or they may be alerts. The abstract factory class contains abstract methods createTsunamiAlert() and createEarthquakeAlert(). The abstract factory is extended by two concrete factories, AlertFactory and WarningFactory. createTsunamiAlert() returns a TsunamiAlert object, and createEarthquakeAlert() returns an EarthquakeAlert object. Both AbstractTsunamiAlert and AbstractEarthquakeAlert are abstract classes extended by {Tsunami|Earthquake}{Alert|Warning} classes.

Builder

SMS implements the builder pattern through the classes SocketDirector, AbstractSocketBuilder, SocketBuilder and SocketWrapper. SocketDirector provides an interface to AbstractSocketBuilder's buildSocket() method through constructSocket(). constructSocket() returns the product, an instance of SocketWrapper, through SocketBuilder's getSocket() method.

Prototype

SMS implements the prototype pattern through AbstractSeismographState and its subclass SeismographState containing the clone method. This method is used because AbstractSeismographState stores a Coordinates object representing a seismograph's coordinates, which do not change over time. The clone method is called by SeismographDecoder to return new instances of the AbstractSeismographState class without having to set the coordinate field of these instances.

Singleton

SMS implements the singleton pattern by instantiating one SocketBuilder object. SocketBuilder class contains a static method getSocketBuilder() which returns the singleton instance of SocketBuilder. It is necessary to implement SocketBuilder as a singleton as it must keep track of free ports.

Adapter

SMS implements adapter through SMSClient, SeismometerListViewTarget, SeismometerListViewAdapter and ListView<T>. SeismometerListViewTarget provides an interface to SMSClient through SeismometerListViewAdapter to update the contents of ListView<T> in a familiar way by simply passing a list of Strings to abstract method setItems().

Composite

SMS implements the composite design pattern through PopulationCentre objects being composed of other PopulationCentre objects. Because a PopulationCentre objects may represent a neighbourhood, city, province/state, region or country, a PopulationCentre representing a country is composed of multiple regions, which are composed of multiple provinces, which are composed of multiple cities, which are composed of multiple neighbourhoods.

Decorator

SMS implements the decorator design pattern through classes MapVisualizer, SeismometerComponent, PopulationCentreComponent along with abstract class MapDecorator and its subclasses TopographicMap and PoliticalMap. The class MapVisualizer is abstract and SeismometerComponent, PopulationCentreComponent and MapDecorator extend MapVisualizer directly. MapDecorator contains reference to MapVisualizer objects and its subclasses draw these objects over topographic or political maps.

Flyweight

SMS implements the flyweight pattern through various objects containing reference to PopulationCentre objects. PopulationCentre objects represent flyweights, and the PopulationCentreFactory represents the flyweight factory. PopulationCentreFactory returns PopulationCentre objects through the getPopulationCentre(String key) method, where key is the name of a population centre. Names of population centres are stored as static String constants in the PopulationCentreFactory class.

Iterator

SMS implements the iterator pattern through SeismographStream storing a list of AbstractSeismographStates and containing a method getNextState() to return the next AbstractSeismographState, getState(int i) to return the AbstractSeismographState at index i in the list, and getFirstState() to return the first AbstractSeismographState in the list.

State

SMS implements the state design pattern through the StateReceiver, AbstractState, SevereState, ModerateState, MinorState and NormalState classes. StateReceiver provides an interface to AbstractState's methods by storing an instance of an AbstractState object. AbstractState is an abstract class which SevereState, ModerateState, MinorState and NormalState extend. AbstractState objects contain reference to their respective StateReceiver class, which in turn contains reference to its current AbstractState object. AbstractState objects may alter the current state of their StateReceiver objects through downgrade() and receiveState().

Template Method

SMS implements the template method design pattern though both AbstractEarthquakeAlert and AbstractHurricaneAlert along with their subclasses. AbstractEarthquakeAlert and AbstractHurricantAlert provide encode() methods, which call abstract methods getCodeLength(), getAlertType() and getSeverity().

Visitor

SMS implements the visitor pattern through AbstractPopulationCentreVisitor visiting a PopulationCentre object and each PopulationCentre object referenced within it. AbstractPopulationCentreVisitor has three concrete subclasses: CounterPopulationCentreVisitor, CollectorPopulationCentreVisitor and ConcretePopulationCentreWarner, which sum the populations of, collect the names of, and set the current state of all PopulationCentres, respectively. AbstractPopulationCentreVisitor contains methods visitLeafPopulationCentre() and visitNodePopulationCentre(). Leaf PopulationCentres are PopulationCentre objects which contain no children, whereas node PopulationCentres contain children.