**GOF Patterns**

**Rodrigo Monteiro Suzana 63069**

**(suzana2314)**

**Template Pattern**

This pattern is located in the following package: *package net.sf.freecol.common.model*

public abstract class NationType extends FreeColSpecObjectType

public class IndianNationType extends NationType

public class EuropeanNationType extends NationType

The rationale for identifying this as a template method pattern instantiation lies in the clear structure of having an abstract class (NationType) providing the template or skeleton for behavior and concrete subclasses (IndianNationType, EuropeanNationType) implementing specific details.

**A diagram of a group of text

Description automatically generated**

**Factory pattern**

This pattern is located in: *package net.sf.freecol.common.resources.ResourceFactory*

**A diagram of a source

Description automatically generated**

This pattern allows for the creation of different types of objects (resources in this case) without explicitly specifying their classes. The method createResource acts as a factory by creating and returning the appropriate subclass of the Resource class based on the type of URI provided. Bellow is the referenced method:

public Resource createResource(String key, String cachingKey, URI uri) {

        final Resource r = resources.get(uri);

        if (r != null) {

            return r;

        }

        final String pathPart;

        if (uri.getPath() != null) {

            pathPart = uri.getPath();

        } else if (uri.toString().indexOf("!/") >= 0) {

            pathPart = uri.toString().substring(uri.toString().indexOf("!/") + 2);

        } else {

            pathPart = null;

        }

        try {

            final Resource resource;

            if ("urn".equals(uri.getScheme())) {

                if (uri.getSchemeSpecificPart().startsWith(ColorResource.SCHEME)) {

                    resource = new ColorResource(cachingKey, uri);

                } else if (uri.getSchemeSpecificPart().startsWith(FontResource.SCHEME)) {

                    resource = new FontResource(cachingKey, uri);

                } else {

                    logger.log(Level.WARNING, "Unknown urn part: " + uri.getSchemeSpecificPart());

                    return null;

                }

            } else if (pathPart.endsWith("\"") && pathPart.lastIndexOf('"', pathPart.length()-1) >= 0) {

                resource = new StringResource(cachingKey, uri);

            } else if (pathPart.endsWith(".faf")) {

                resource = new FAFileResource(cachingKey, uri);

            } else if (pathPart.endsWith(".sza")) {

                resource = new SZAResource(cachingKey, uri);

            } else if (pathPart.endsWith(".ttf")) {

                resource = new FontResource(cachingKey, uri);

            } else if (pathPart.endsWith(".wav")) {

                resource = new AudioResource(cachingKey, uri);

            } else if (pathPart.endsWith(".ogg")) {

                if (pathPart.endsWith(".video.ogg")) {

                    resource = new VideoResource(cachingKey, uri);

                } else {

                    resource = new AudioResource(cachingKey, uri);

                }

            } else if (key.startsWith("sound.")) {

                resource = new AudioResource(cachingKey, uri);

            } else {

                resource = new ImageResource(cachingKey, uri);

            }

            resources.put(uri, resource);

            return resource;

        } catch (IOException ioe) {

            logger.log(Level.WARNING, "Failed to create " + uri, ioe);

            return null;

        }

    }

**Observer Pattern**

This pattern is located in the following package: *net.sf.freecol.client.gui.panel*

Respectively in the following classes: TransactionListener, EuropePanel (TransactionLog is inside this class in line 509) and Market.

A diagram of a diagram

Description automatically generated with medium confidence

The TransactionListener interface serves as a contract, providing a way for the TransactionLog class to be notified when changes occur in the class Market. When relevant changes take place within the Market class, it triggers notifications to all subscribed TransactionListeners. This mechanism allows the observers to react to changes in the subject's state.

**Market**

    /\*\*

     \* Adds a transaction listener for notification of any transaction

     \*

     \* @param listener The {@code TransactionListener} to add.

     \*/

    public void addTransactionListener(TransactionListener listener) {

        transactionListeners.add(listener);

    }

    /\*\*

     \* Removes a transaction listener

     \*

     \* @param listener The {@code TransactionListener} to remove.

     \*/

    public void removeTransactionListener(TransactionListener listener) {

        transactionListeners.remove(listener);

    }

    /\*\*

     \* Gets the listeners added to this market.

     \*

     \* @return An array of all the {@code TransactionListener}s

     \*     added, or an empty array if none found.

     \*/

    public TransactionListener[] getTransactionListener() {

        return transactionListeners.toArray(new TransactionListener[0]);

    }

**TransactionListener**

public interface TransactionListener {

    /\*\*

     \* Logs a purchase

     \*

     \* @param goodsType The type of goods which have been purchased

     \* @param amount The amount of goods which have been purchased

     \* @param price The unit price of the goods

     \*/

    public void logPurchase(GoodsType goodsType, int amount, int price);

    /\*\*

     \* Logs a sale

     \*

     \* @param goodsType The type of goods which have been sold

     \* @param amount The amount of goods which have been sold

     \* @param price The unit price of the goods

     \* @param tax The tax which has been applied

     \*/

    public void logSale(GoodsType goodsType, int amount, int price, int tax);

}

**TransactionLog**

  /\*\*

         \* Initializes this TransactionLog.

         \*/

        public void initialize() {

            getMyPlayer().getMarket().addTransactionListener(this);

            setText("");

        }

        /\*\*

         \* Cleans up this TransactionLog.

         \*/

        public void cleanup() {

            getMyPlayer().getMarket().removeTransactionListener(this);

        }

// Implement TransactionListener

        @Override

        public void logPurchase(GoodsType goodsType, int amount, int price) {

            int total = amount \* price;

            StringTemplate t1 = StringTemplate.template("europePanel.transaction.purchase")

                .addNamed("%goods%", goodsType)

                .addAmount("%amount%", amount)

                .addAmount("%gold%", price);

            StringTemplate t2 = StringTemplate.template("europePanel.transaction.price")

                .addAmount("%gold%", total);

            add(Messages.message(t1) + "\n" + Messages.message(t2));

        }

        @Override

        public void logSale(GoodsType goodsType, int amount,

                            int price, int tax) {

            int totalBeforeTax = amount \* price;

            int totalTax = totalBeforeTax \* tax / 100;

            int totalAfterTax = totalBeforeTax - totalTax;

            StringTemplate t1 = StringTemplate.template("europePanel.transaction.sale")

                .addNamed("%goods%", goodsType)

                .addAmount("%amount%", amount)

                .addAmount("%gold%", price);

            StringTemplate t2 = StringTemplate.template("europePanel.transaction.price")

                .addAmount("%gold%", totalBeforeTax);

            StringTemplate t3 = StringTemplate.template("europePanel.transaction.tax")

                .addAmount("%tax%", tax)

                .addAmount("%gold%", totalTax);

            StringTemplate t4 = StringTemplate.template("europePanel.transaction.net")

                .addAmount("%gold%", totalAfterTax);

            add(Messages.message(t1) + "\n" + Messages.message(t2)

                + "\n" + Messages.message(t3) + "\n" + Messages.message(t4));

        }

    }