COMP 3004 – SCAPES Deliverable #1

Design Pattern Presentation

Team 36-Backslash-n

Pallab Saha

Redwan Wadud

Salim

Robby

Submitted to: Dr. Christine Laurendeau

COMP 3004 – Object-Oriented Software Engineering

School of Computer Science Carleton University

Bridge Design Pattern

- We have chosen to use Bridge design pattern to abstract the Repository functionality

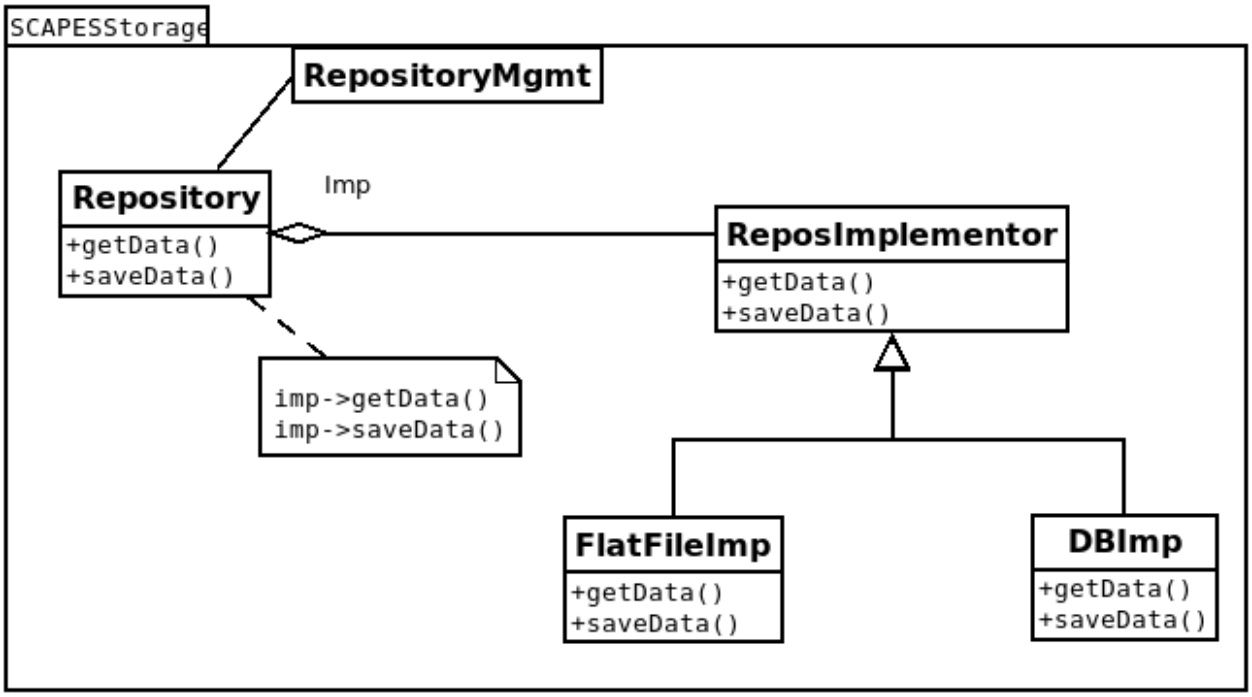
- Bridge Design Pattern
    - decouples Abstraction (Interface class) from Its implementation (Implementor classes)

    - Hides implementation details from clients

    - Abstraction (Interface Object) forwards client requests to its Implementor Object.

    - Changing an implementation has no impact to the Clients.

    - Allows improved extensibility. Abstraction and Implementation hierarchies can be extended independently.



- In Deliverable #1, we have implemented FlatFile Implementor as a concreteImplementor.

- In future, we would extend the Repository Implementor. We can add another concrete implementor, e.g. Database based Implementor.

- The future change of the implementor will not require any change from clients.

```
// Interface Class
class Repository
{
public:
   Repository(RepositoryImplementor *imp);
   bool getSourceData(QString filename, QString *outData, QString *outErrTxt);
   bool setSourceData(QString filename, QString *inData, QString *outErrTxt);
private:
   RepositoryImplementor   *imp;
};
// The concrete Implementor(FlatFile) object is bound to the interface
// (Repositority) object
Repository::Repository(RepositoryImplementor *impRef)
{
   imp = impRef;
}
// GetSourceData and SetSourceData operations are delegated to the Concrete
 // Implementor object, which does the actual work.
bool Repository::getSourceData(QString filename, QString *outData, QString *outErrTxt)
{
   return imp->getSourceData(filename, outData, outErrTxt);
}
bool Repository::setSourceData(QString filename, QString *inData, QString *outErrTxt)
{
   return imp->setSourceData(filename, inData, outErrTxt);
}
```

```
// Virtual Implementor class
class RepositoryImplementor
{
public:
    RepositoryImplementor();
   virtual ~RepositoryImplementor();
    virtual bool getSourceData(QString filename, QString *outData, QString *outErrTxt) = 0;
    virtual bool setSourceData(QString filename, QString *inData, QString *outErrTxt) = 0;
};

// Concrete Implementor
// Currently only FlatFile implementation is done.
// In future, other implementation will be added.
class FlatFileImp : public RepositoryImplementor
{
public:
    FlatFileImp();
    ~FlatFileImp();
    bool getSourceData(QString filename, QString *inData, QString *outErrTxt);
    bool setSourceData(QString filename, QString *outData, QString *outErrTxt);
};
```