**Lab**​​**4**​​**-**​​**Generics,**​​**Exceptions,**​​**Cloning,**​​**and**​​**Adapter**​​**Design**​​**Pattern**

This ​ lab​ ​ contains​ ​ in-class​ ​ exercises​ ​ ​related ​ to​ ​ generics,​ ​ exception,​ ​ and​ ​ cloning.​ ​ Optional​ ​ task​ ​ 4:​ ​ Adapter​ ​ Design​ ​ Pattern​

**Task** ​ **1:**​​ Given ​ the​ ​ class​ ​Pair<F,​ ​S>

public​ ​class​ ​Pair<F,​ ​S>​ ​{

​ private​ ​ ​F​ first;​

​ ​private​ ​S​ ​second;

​ ​/\*\*

​ ​\*​ ​Constructor​ ​of​ ​Pair​ obj​

​ ​\*

​ ​\*​ ​@param​ ​f​ ​object​ ​of ​ type​ ​ F​

​ ​\* ​ @param​ ​ ​s​ object​​ ​of​ ​type​ ​S

​ ​\*/

​ ​public​ ​Pair(F​ ​f,​ ​S​ ​s)​ {​

​ ​first​ ​=​ ​f;

​ second​ ​ =​ ​ s;​

​ ​}

​ ​/\*\*

​ ​\*​ ​Print ​ the​ ​ Pair​ ​ object​

​ ​\*

​ ​\*​ ​@return​ string​ ​ ​of​ ​Pair​ ​representation

​ ​\*/

​ ​public​ ​String​ ​toString()​ ​{

​ return​ ​ "("+​ ​ first​ ​ +​ ​ ",​ ​ ​" ​ +​​ second​ ​ ​+ ​​")";

​ ​}

​ ​/\*\*

​ ​\*​ ​Flips​ ​the​ ​Pair​ ​obj​ ​elements

​ ​\*​ ​for​ ​example​ ​the​ ​pair​ ​(a,​ ​b)​ ​becomes​ ​(b,​ ​a)

​ ​\*

​ ​\*​ ​@param​ ​p​ ​object ​ of​ ​ type​ ​ Pair​

​ ​\*​ @return​ ​ an​​ ​object​ ​of​ ​type​ Pair​ ​​with ​​its

​ ​\*​ ​components ​ flipped​

​ ​\*/

​ ​public​ ​static ​ /\*​ ​ ???​ ​ \*/​ ​ flip(​ ​ /\*​ ​ ​??? ​ \*/)​​ {​

​ ​/\*​ ​???​ ​\*/

​ ​}

​ /\*\*​

​ ​\*​ ​Entry ​ point​

​ ​\*​ ​@param​ ​args​ ​array​ ​of​ ​Strings

​ ​\*/

​ ​public​ ​static​ ​void​ ​main(String[]​ ​args)​ ​{

​ ​Pair<Integer,​ ​String>​ ​p​ ​=​ ​new​ ​Pair<>(1,​ ​"Test");

​ ​System.out.println(p);

​ System.out.println(Pair.flip(p));​

​ ​}

}

Implement ​ the​ ​ method​ ​​ ​public​ ​static​ ​/\*​ ​???​ ​\*/​ ​flip(​ /\*​​ ​??? ​​\*/) such ​ as​ ​ for​ ​ the​ ​ parameter​ ​(1,​ ​"Test") ​ the ​ result​ ​ will​ ​ be​ ​ (​ "Test", ​ 1)​ Clone ​ an​ ​ object​ ​ of​ ​ type​ ​ Pair.​

**Task** ​ **2:**​

Write ​ a​ ​ class​ ​ that​ ​ represents​ ​ a​​ matrix​ ​ of​ ​ ​bytes

public​ ​class​ ​Matrix​ ​{

​ ​private​ byte[][]​ ​ element;​ ​ //matrix​ ​ values​

​ ​private​ ​int​ ​nrows, ​​ncols;​ ​//number​ ​of​ ​rows​ ​and​ ​number​ ​of​ ​columns

​ ​//create ​ a​ ​ matrix​ ​ of​ ​ size​ ​ that​ ​ ​has ​ given​ ​ ​dimensions

​ ​public​ ​Matrix(int​ ​nrows,​ ​int​ ​ncols)​ ​{

​ ​}

​ ​//create​ ​a​ ​matrix​ ​with​ ​values​ ​from​ ​another​ ​matrix

​ ​public​ ​Matrix(Matrix​ ​source)​ ​{

​ ​}

​ ​//create​ ​a​ ​matrix​ ​from​ ​array​ ​of​ ​array​ ​of​ ​bytes

​ ​public​ ​Matrix(byte[][] ​ b)​ ​ {​

​ ​}

​ ​//add​ ​this​ ​matrix​ ​to​ ​a​ ​second​ ​matrix ​ and​ ​ return​ ​ the​​ ​sum

​ ​//​ ​IllegalMatrixDimensionException​ ​is​ ​user​ ​defined​ exception​

​ ​public​ ​Matrix ​​add(Matrix​ ​second)​ ​throws​ ​IllegalMatrixDimensionException​ ​{ ​ ​}

​ ​//get​ ​the​ ​value​ ​of​ ​this​ ​matrix​ ​at​ ​row​ ​r​ ​and​ ​column​ ​c

​ //OutOfRangeMatrixIndexException​ ​ is​ ​ user​ ​ defined​ ​ exception​

​ ​public​ ​byte​ ​val(int​ ​r,​ ​int​ ​c)​ ​throws​ ​OutOfRangeMatrixIndexException​ ​{ ​ ​}

​ ​//set ​ the​ ​ element​ ​ ​value ​ ​val ​ at​ ​ row​ ​ r​ ​ and​ ​ column​​ c​

​ ​public​ ​void​ ​setElement(byte​ ​val,​ ​int​ ​r,​ ​int​ ​c)​ ​throws

​ ​OutOfRangeMatrixIndexException​ ​{}

​ //implement​ ​ ​toString,​ ​equals,​ ​hashCode

​ ​//returns​ ​the​ ​maximum ​ ​value​ of​ ​ this​ ​ matrix​

​ ​public​ ​static​ ​byte​ ​max()​ ​{

​ ​} ​ ​//test​ ​your​ ​class​ ​methods

​ ​public​ ​static​ ​void​ ​main(String[]​ ​args)​ ​{

​ ​}

}

**Task** ​ **3**​​**:**​Develop ​ a​ ​ class​​Matrix<T> ​​ of​ ​ a​ ​ generic​ ​ type​​ ​​T

**Task** ​ **4:**​​ **(**​ **optional):**

1. In the adapter example provided, change the ​**Inscribable** interface, so that the method is ​ defined​ ​ by:​

public​ interface​ ​Inscribable​ ​{

​ */\*\**

​​*\**​​*Calculates*​​*the*​​*area*​​*of*​​*a*​​*circle*​​*inscribed*​​*in*​​*a*​​*square* ​​*\**

​​*\**​ *@param*​​*width*​​*The*​​*dimension*​​*of*​​*the*​​*square*

​​*\**​ *@return*​​*The*​​*area*​​*of*​​*the*​​*circle*​​*inscribed*​​*in*​​*the*​​*Square*

​​*\*/*

​ double​ circleArea​(double​ ​ *width*​);

}

What else needs to be changed? Refactor all the adapter components: the ​Adaptee class​ and ​ its​ ​ interface,​ ​ the​ ​Adapter ​ class ​ and​ ​ ​the ​Client ​ class.

1. ​ Develop​ ​ an​ ​ Adapter​ ​ pattern​ ​ implementation​ ​ for​​ solving​ ​ the​ ​ following​ ​ problem:​

The client wants to add two numbers in the ​Binary format,​ where Binary​ is a class that you have to implement​. However,​ there exists an implementation that adds two numbers as ​ integers.​

For example, if your client invokes add​ (Binary x, Binary y), you have an implementation ​ of​ ​add​ ​(Integer​ ​x,​ ​Integer​ ​y)

1. ​ Design​ ​ and​ ​ implement​ ​ a​ ​ solution​ ​ for​​ the​ ​ following​ ​ problem​ ​ statement:​

Let us suppose there is a web commerce application that uses a gateway payment system. The current gateway uses a representation of the credit card date in the format year-month-day.

For some internal reasons, management has decided to replace the gateway with another

one ​ that​ ​ uses​ ​ a​ ​ different​ ​ representation​​ format,​ ​ such​ ​ as:​ ​day/month/year

Hint:

Apply ​ the​ ​ adapter​ ​ design​ ​ pattern​ ​ for​​ mapping​ ​ the​ ​ old​​ ​gateway ​ format​ ​ to​​ the​ ​ new​​ one​