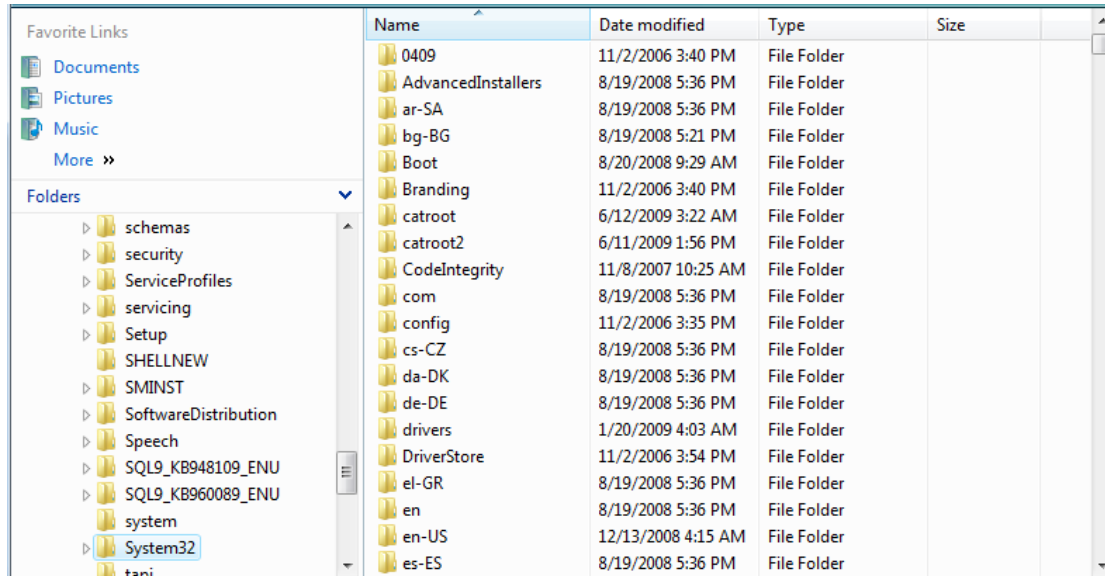


פרק 1. בעיות תכנון

שאלה 1. ניהול קבצים (50 נקודות)

נתונה מערכת לניהול קבצים **WindowsExplorer**. למערכת יש ממשק גרפי שמורכב משני חלקים: שמאל וימין. החלק השמאלי מציג את עץ הספריות ואילו החלק הימני מציג את רשימת הקבצים הנמצאים בספרייה שנבחרה בחלק השמאלי.



Name	Date modified	Type	Size
0409	11/2/2006 3:40 PM	File Folder	
AdvancedInstallers	8/19/2008 5:36 PM	File Folder	
ar-SA	8/19/2008 5:36 PM	File Folder	
bg-BG	8/19/2008 5:21 PM	File Folder	
Boot	8/20/2008 9:29 AM	File Folder	
Branding	11/2/2006 3:40 PM	File Folder	
catroot	6/12/2009 3:22 AM	File Folder	
catroot2	6/11/2009 1:56 PM	File Folder	
CodeIntegrity	11/8/2007 10:25 AM	File Folder	
com	8/19/2008 5:36 PM	File Folder	
config	11/2/2006 3:35 PM	File Folder	
cs-CZ	8/19/2008 5:36 PM	File Folder	
da-DK	8/19/2008 5:36 PM	File Folder	
de-DE	8/19/2008 5:36 PM	File Folder	
drivers	1/20/2009 4:03 AM	File Folder	
DriverStore	11/2/2006 3:54 PM	File Folder	
el-GR	8/19/2008 5:36 PM	File Folder	
en	8/19/2008 5:36 PM	File Folder	
en-US	12/13/2008 4:15 AM	File Folder	
es-ES	8/19/2008 5:36 PM	File Folder	

המשתמש יכול להוסיף, למחוק או לשנות שם של קובץ דרך כל צד בממשק. עץ הספריות או רשימת הקבצים. עץ הספריות מציג אך ורק ספריות ואילו רשימת הקבצים מציגה גם קבצים וגם ספריות

- במידה והמשתמש הגדיר ספרייה חדשה דרך העץ (חלק שמאלי), הספרייה חייבת להופיע מיד גם ברשימת הקבצים בחלק ימין והפוך.
- במידה והמשתמש מחק ספרייה דרך העץ (חלק שמאלי), הספרייה חייבת להימחק מיד גם מרשימת הקבצים בחלק ימין והפוך.
- במידה והמשתמש שינה את שמה של הספרייה דרך העץ (חלק שמאלי), שם של הספרייה חייב להשתנות מיד גם ברשימת הקבצים בחלק ימין והפוך.
- במידה והמשתמש הוסיף, מחק או שינה שם של קובץ רגיל דרך רשימת הקבצים, אין זה משפיע על המוצג בעץ הספריות

הנח/י כי חלק שמאל של הממשק ממומש ע"י מחלקה **DirTree** וחלק ימין ממומש ע"י מחלקה **FileList**. שתי המחלקות מצביעות על אותה רשימה של קבצים.

מחלקה **FileList** כוללות מתודות הבאות:

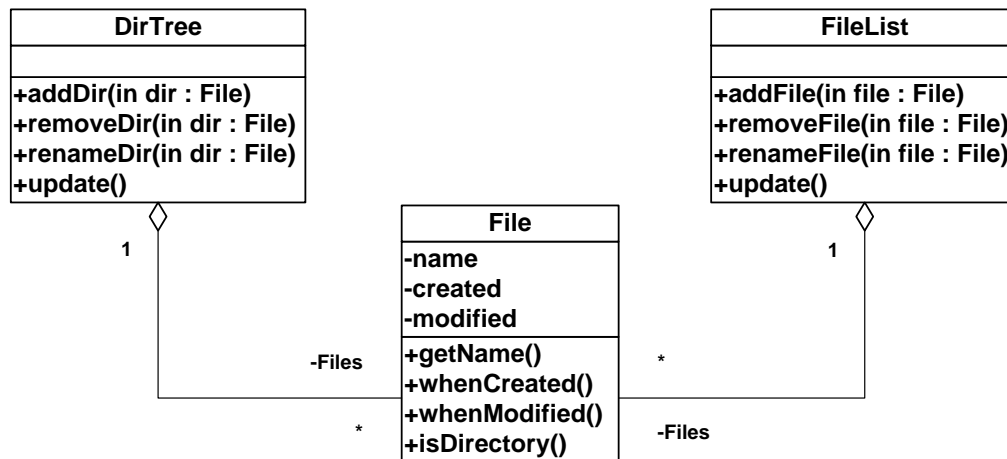
- **void addFile (File _file)** - מוסיפה קובץ לרשימת הקבצים
- **void removeFile (File _file)** – מוחקת קובץ מרשימת הקבצים.
- **void renameFile (File _file)** – משנה שם של קובץ.
- **void update()** – מציירת את עצמה מחדש על בסיס רשימת הקבצים



מחלקה DirTree כוללת מתודות הבאות:

- void addDir (File _dir) - מוסיפה ספרייה לרשימת הקבצים
- void removeDir (File _dir) – מוחקת ספרייה מרשימת הקבצים.
- void renameDir (File _dir) – משנה שם של ספרייה.
- void update() – מציירת את עצמה מחדש על בסיס רשימת הקבצים

המחלקה File מייצגת ספרייה וכוללת שדות כמו שם הקובץ, סימון של ספרייה, תאריך יצירה וכו'



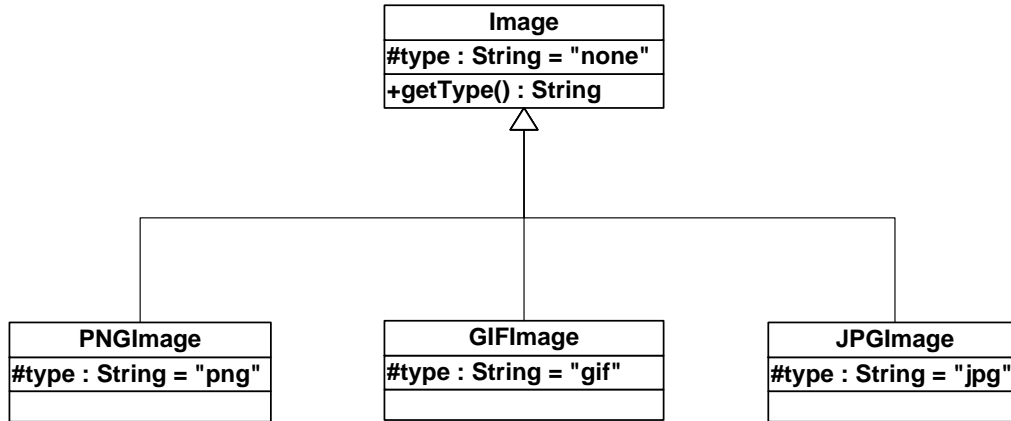
עליך לתכנן מנגנון המסנכרן בין שני החלקים.

- תכנת/י מחלקה אבסטרקטית **Observer** בעלת מתודה אחת בלבד – **void update()**. שנה מחלקות **FileList** ו-**DirTree**, כך שיהיו מחלקות יורשות של **Observer**. אין צורך לממש את המתודות. כתב/י רק את ההגדרות (10 נקודות).
- תכנת/י מחלקה **FileSynchronizer** שמטרתה לסנכרן בין שני חלקיו של הממשק (15 נקודות):
 - בחר מבנה נתונים המתאים לבעיה. אפשר להשתמש בכל מבנה נתונים בסיסי, כגון **Vector**, **Stack**, **HashTable** או **List**. הסבר את הבחירה ואופן השימוש במבנה נתונים
 - מתודות ניהוליות, כגון קונסטרקטור ודיסטרקטור – ממשו את המחלקה כסינגלטון
 - הגדר וממש מתודה **void update()**, שמטרתה להודיע לשני החלקים על השינוי
- ממשי/י מתודה **addDir** של המחלקה **DirTree**. תתעלמי/י מקוד שקשור להוספה של ספרייה ותתמקדי/י בקוד שקשור לסנכרון (5 נקודות)
- ממשי/י מתודה **removeFile** של המחלקה **FileList**. תתעלמי/י מקוד שקשור למחיקה של קובץ ותתמקדי/י בקוד שקשור לסנכרון (5 נקודות)
- צייר/י את תרשים המחלקות של הפתרון. אין צורך לפרט מתודות ומשתנים של המחלקות (5 נקודות)
- צייר/י את תרשים סדרתי (sequence) שיתאר אופי השימוש במנגנון הסנכרון. עליך להראות הידברות בין האובייקטים: **FileList**, **DirTree** ו-**FileSynchronizer** (10 נקודות)



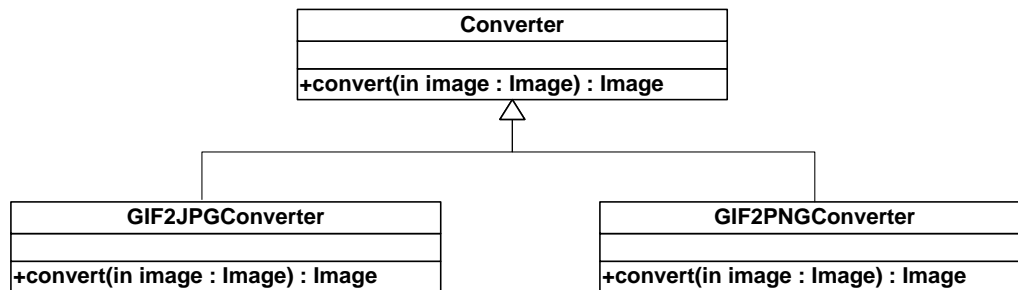
שאלה 2. המרת פורמטים של תמונות (50 נקודות)

נתונה מערכת לניהול תמונות `ImageCenter`. המערכת מסוגלת לעבוד עם תמונות בפורמטים שונים, כגון `jpg`, `gif`, `png`, `bmp` ואחרים. הנח/י כי כל תמונה מיוצגת ע"י מחלקה לפי היררכיה הבאה:



- מתודה `String getType()` מחזירה פורמט של התמונה. למשל, במחלקה `GIFImage` המתודה תחזיר `"gif"`.

אחת היכולות החשובות של המערכת היא המרה בין פורמטים שונים. ההמרה נעשית ע"י ממירים (converters) כשכל ממיר יודע לטפל בזוג פורמטים בלבד. למשל, ממיר `gif`-ל-`jpg` יודע להמיר תמונות `gif` לתמונות `jpg` והפוך, אבל לא יודע להמיר למשל תמונות `gif`-ל-`png`. ניתן להוסיף ממירים חדשים.



עליך לתכנן מחלקה האחראית על הממירים (`ConverterMgr`) השונים. המחלקה חייבת לתמוך ב

- הוספת ממירים חדשים.
- לספק למערכת ניהול את הממיר המתאים. אם הממיר הדרוש אינו נתמך, זרוק `NotSupportedException`

א. בחר/י מבנה נתונים המתאים לאחסון ממירים. אפשר להשתמש בכל מבנה נתונים בסיסי, כגון `Vector`, `Stack`, `HashTable` או `List`. הסבר את הבחירה ואופן השימוש במבנה נתונים (5 נקודות)

ב. כתב/י את ההגדרות של מחלקה `ConverterMgr`. הגדרות צריכות לכלול (10 נקודות):

- מתודה `addConverter`, תחשבו על הפרמטרים הדרושים והטיפוס המוחזר
- מתודה `getConverter`, על הפרמטרים הדרושים והטיפוס המוחזר
- מתודות ניהוליות, כגון `קונסטרקטור` ודיסטרוקטור – ממשו את המחלקה כסינגלטון



- ג. תכנת/י מחלקה **NotSupportedException** אשר תגזר ממחלקה **exception** של **C++** (5 נקודות)
- ד. ממש/י את המתודות של **ConverterMngr** שהגדרת (15 נקודות)
- ה. צייר/י את תרשים שיתוף פעולה (collaboration) שיתאר השימוש במחלקה **ConverterMngr**. עליך לתאר הידברות בין 4 אובייקטים (15 נקודות):
- **ImageCenter** המחלקה הראשית של התוכנית
 - **Image** – תמונה כלשהי
 - **ConverterMngr**
 - **Converter** – ממיר תמונות



// Converter Mngl. cpp

(3)

```
Converter Mngl * Converter Mngl:: _inst = 0;
```

```
Converter Mngl * Converter Mngl:: get_instance ()
```

```
{
```

```
    if (_inst == 0)
```

```
        _inst = new Converter Mngl();
```

```
    return _inst;
```

```
}
```

```
void Converter Mngl:: add Converter (string f1,  
                                     string f2, Converter * conv)
```

```
{
```

```
    string key1 = f1 + "2" + f2;
```

```
    string key2 = f2 + "2" + f1;
```

```
    _convs[key1] = conv;
```

```
    _convs[key2] = conv;
```

```
}
```



(7)

(1)

```

class Observer
{
public:
    virtual void update() = 0;
}

class FileList : public Observer
{
public:
    void addFile (File file);
    void removeFile (File file);
    void renameFile (File file);
    void update ();
}

class DirTree : public Observer
{
public:
    void addDir (File dir);
    void removeDir (File dir);
    void renameDir (File dir);
    void update ();
}

```



(1)

```
// File Synchronizer.h
class File Synchronizer
```

```
{
  Vector< Observer* > _observers;
  File Synchronizer() {}
  ~File Synchronizer() {
    _observers.clear();
  }
  static File Synchronizer* _instance;
```

```
public:
  static File Synchronizer* get_instance() {
    if (_instance == 0)
      _instance = new File Synchronizer();
    return _instance;
  }
  void add Observer (Observer* _observer);
  void remove Observer (Observer* _observer);
  void update();
}
```



// File Synchronizer.cpp

File Synchronizer → FileSynchronizer::_instance = 0;

```
void FileSynchronizer::addObserver ( Observer * _observer )  
{  
    _observers.push_back ( _observer );  
}
```

```
void FileSynchronizer::removeObserver ( Observer * _observer )  
{  
    vector ( Observer * )::iterator iter;  
    iter = find ( _observers.begin(), _observers.end(),  
                _observer );  
    if ( iter != _observers.end )  
        _observers.erase ( iter );  
}
```

```
void FileSynchronizer::update ()  
{  
    Observer * temp;  
    for ( int i = 0; i < _observers.size(); i++ ) {  
        temp = _observers[i];  
        temp → update();  
    }  
}
```



3

1

```

DirTree::addDir (File -dir) {
  // add File to the list
  FileSynchronizer * synch =
    FileSynchronizer::get_instance();
  synch -> update();
}

```

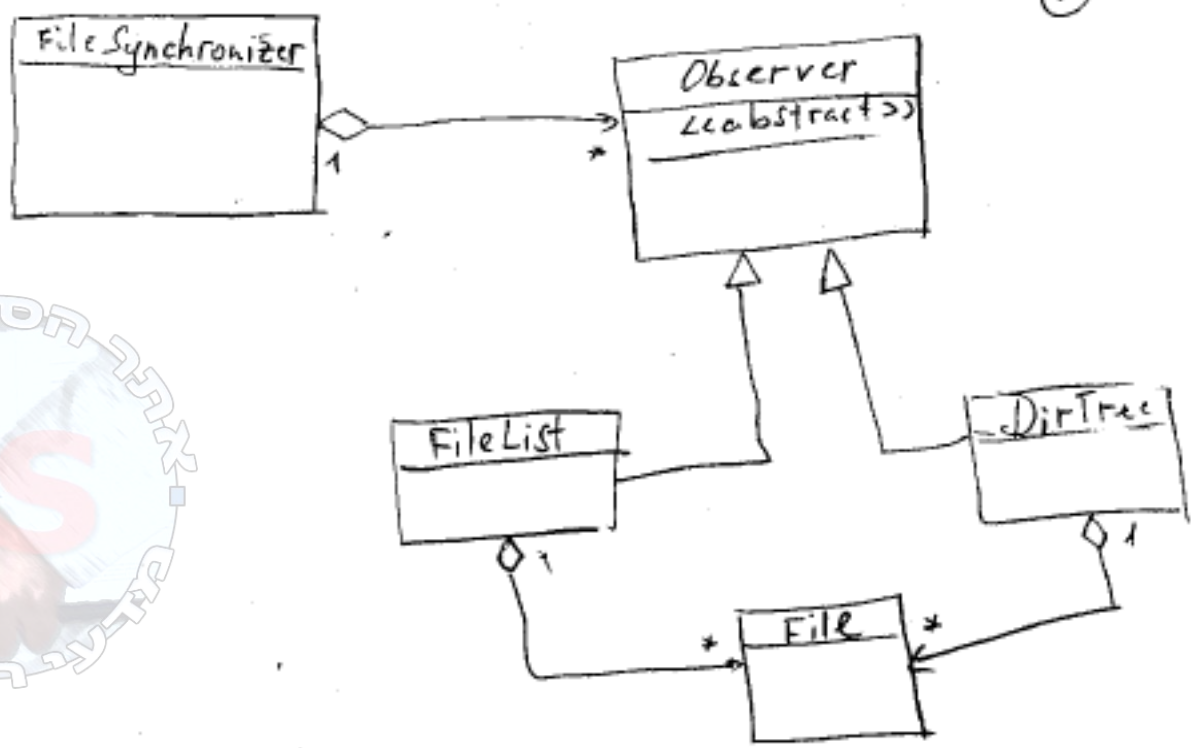
2

```

void FileList::removeFile (File -file) {
  // remove file from the list
  if (!file.isDirectory()) return;
  FileSynchronizer * synch =
    FileSynchronizer::get_instance();
  synch -> update();
}

```

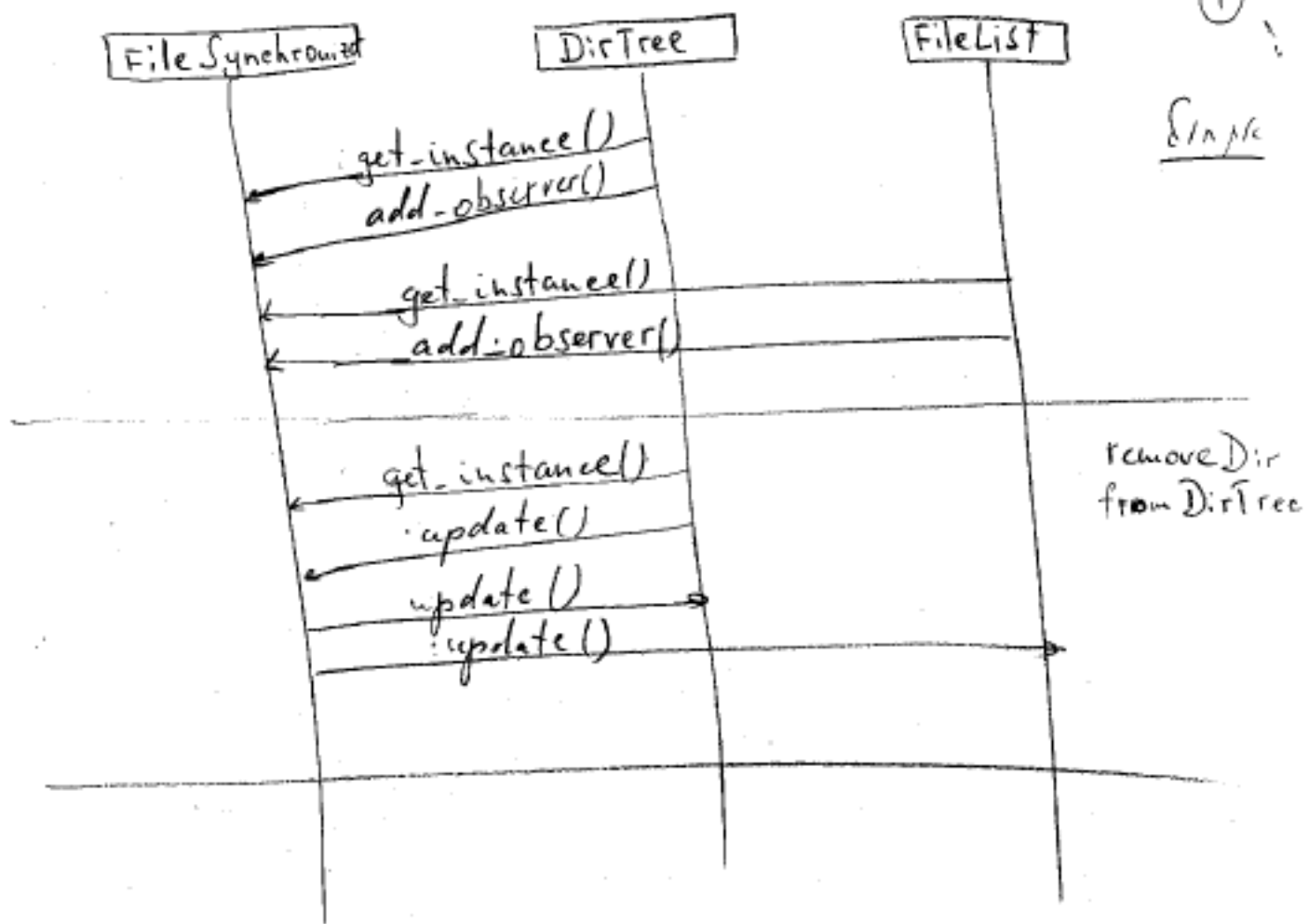
3



4

①

Simple



⑤

Map ← (Hash Table) אגז'ע אפאט (1)

key = "png2gif"
key = "gif2png"

key	Converter*

אגז'ע אפאט
אגז'ע אפאט - אגז'ע אפאט
אגז'ע אפאט אגז'ע אפאט

```

class ConverterMgr {
    map<String, Converter*> -convs;
    ConverterMgr() {}
    ~ConverterMgr() { -convs.clear; }
    static ConverterMgr* -inst;

public:
    static ConverterMgr* get_instance();
    void addConverter (String format1, String format2,
                      Converter* convert);
    Converter* getConverter (String format1, String format2)
    throws NotSupportedConverterException
}

```

```

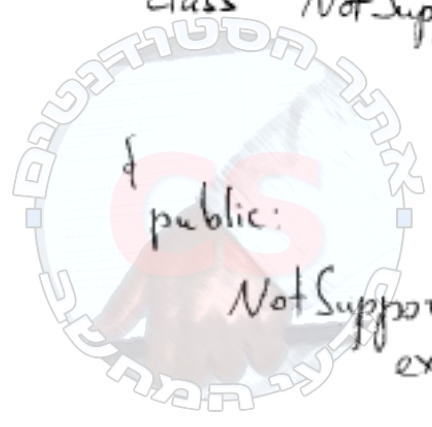
class NotSupportedConverterException :
    public exception

```

```

NotSupportedConverterException():
    exception ("converter is not supported") {}
}

```



```

! Converter* getConverter (string f1, string f2) throws
    NotSupportedException
{
    string key = f1 + "2" + f2;

```

```

    map<string, Converter* >::Iterator iter,
    iter = _convr.find(key);
    if (iter == _convr.end())
        throw NotSupportedException;
    return iter->second;
}

```

Convert(image, format)

