Hacettepe University

Department of Computer Science & Engineering

BBM104 Programming Lab.

Assignment 2

Subject **:** Inheritance & Polymorphism

Due Date **:** 04.08.2014 23:59

Programming Language : Java SE 1.7

Advisors **:** Assist. Prof. Sevil Şen, R.A. Yasin Şahin

1. INTRODUCTION

Inheritance

In general, different kinds of objects often have a certain amount in common with each other. We can observe that all over the world. Students, Lecturers and Officers, for example, all have common characteristics caused being a human such as age, address or national identification number.

A class that is derived from another class is called a *subclass* (also a *derived class, extended class,* or *child class*). The class from which the subclass is derived is called a *superclass* (also a*base class* or a *parent class*). In previous example, *Student* is subclass and Human(or Person) is the superclass.

In the context of Java, all the classes are derived from *Object* class. If a class has no user-defined superclass, its superclass will be *Object* class. As *de facto* of programming criteria, reusability and extendibility are the main basic goal of *Inheritance*. A subclass contains all the attributes and abilities of its superclass. So we can say, each object defined in Java, have some common behaviours inherited from *Object* class.

Polymorphism

Polymorphism literally means ``having many forms." Polymorphism is the process of using a method in different ways for different set of inputs given. More precisely, polymorphism is the ability of objects belonging to different types to respond to method calls of the same name, each one according to an appropriate type-specific behavior.

2. EXPERIMENT

In this experiment, you are expected to develope a “Content Management System” –CMS- by using Java programming language. You should handle Roles, Users and Contents within system.

Content:

Three types of contents can be accepted by CMS; Video, Image, Document. Each content type has common members such as id, name, mime-type, size. But also, they show different behaviours in some cases. For example, a video is being streamed according to its frame per secon(fps) attribute.

Document: Document is a basic type for CMS. The only attribute of a document is character count. The number of characters also gives the size of the document. A document content should reported as viewed in CMS when GET command is called.

Image: Image content has only resolution attribute. But resolution attribute consist two integer value. First is pixel size for width and second is pixel size for height of the image.

$$\text{Image size} = 3 * \text{ResolutionWidth} * \text{ResolutionHeight} \qquad \text{Eq.1}$$

Eq.1 gives the size of an image. The constant value *3* in the Eq.1. is about RGB. Means, a pixel has 3 byte (1 byte for each Red, Green and Blue). An image content should be reported as drawed in CMS when GET command is called.

Video: Video content consist of three attributes; FPS (frame per second), Resolution (width x height values), Video Length (in second). Actually, a video is a collection of images. The size of a video(in byte) can be calculated as given Eq.2.

$$\text{Video size} = 3 * \text{FPS} * \text{ResolutionWidth} * \text{ResolutionHeight} * \text{Lenght} \qquad \text{Eq.2}$$

A video content should be reported as streamed in CMS when GET command is called.

Role:

CMS has dynamic role management. There will be three types of operations; PUT, GET and LIST. If a non-authorized user try to save a content, get a content or list contents, CMS informs about permission has been denied.

User:

User objects are the commanders of the CMS. You should check the user's authority whether if it is permitted to run given command before execute command. For example, if a user doesn't have write permission, it can not put a content to the CMS.

CMS Rules:

You should build your data structure to keep objects in it. Do not use java.util.Collection class and any of its subclasses such as ArrayList,Set etc. Do not use any ready-to-use Data Structure class or interface. You are supposed to design and code your own list. Also you can not use array notations(i.e. Object[])

3. INPUT / OUTPUT

Your application should handle PUT – GET – LIST commands given by input file. Two seperate output files should be generated. You can read commands from input.inp file, and write down the results into result.out and log.out files. You are not supposed to handle arguments for this experiment, i/o files names are static as given.

Input (input.inp)

PUT ROLE *ID,NAME,AUTHORITY*

ID : Unique identication number of the role

NAME : Role name.

AUTHORITY : *w* to write(put), *r* to read(get) and *l* to list authorization. If a role provides write and read authorizations, this parameter will be *wr*.

Command Definition : Defines a role to system.

Result Information : *NAME* role is added.

Log Information : ROLE ADDED

PUT USER *ID,FULLNAME,ROLE*

ID : Unique identication number of the user

FULLNAME : User's fullname

ROLE_ID : A reference to role

Command Definition : Defines a user to system.

Result Information : A(n) *ROLE_NAME* is added : *FULLNAME*

Log Information : USER ADDED

PUT CONTENT *ID, NAME, MIME-TYPE,{HEADER_SUMMARY}* REQUESTED_BY *USER_ID*

ID : Unique identication number of the content

NAME : Filename of the content. Be careful about extension separator ".".

MIME-TYPE : Mime-type can be one of these:

Documents' mime-types : TXT, PDF

Videos' mime-types : AVI, MP4

Images' mime-type : JPG, PNG

{HEADER_SUMMARY}: Number of attributes change based on content type as given below:

Documents' attributes :

1) *The number of characters within document*

Videos' attributes :

1) *FPS value*
2) *Resolution*
3) *Video Length*

Images' attributes :

1) *Resolution*

REQUESTED_BY *USER_ID :* Refers to user who execute the command.

Command Definition : Saving operation will be done by this command. Be care about authorization. The requestor has to have writing ability.

Result Information : *USER_FULLNAME* put a video/an image/a document with size of *BYTE_VALUE**

Log Information : *A VIDEO/AN IMAGE/A DOCUMENT* ADDED

**Size of a content can be calculated as*

*Video size = FPS * (3 * ResolutionX * ResolutionY) * Video Length*

Image size = 3 ResolutionX * ResolutionY*

Document size = characters count

GET *CONTENT_ID* REQUESTED_BY *USER_ID*

CONTENT_ID : Refers to content

REQUESTED_BY *USER_ID :* Refers to user who execute the command.

Command Definition : Displaying operation will be done by this command. The requestor has to have reading ability.

Result Information : *CONTENT_NAME streaming/drawing/viewing* by *USER_FULLNAME*

Log Information : *A VIDEO/AN IMAGE/A DOCUMENT* IS READ

LIST BY SIZE REQUESTED_BY *USER_ID*

REQUESTED_BY *USER_ID :* Refers to user who execute the command.

Command Definition : Sorting by size operation will be done by this command. The requestor has to have listing ability.

Result Information : *List of contents returned from sorting algorithm.*

Log Information : *USER_FULLNAME* LISTED BY SIZE

LIST BY TYPE *MIME-TYPE* REQUESTED_BY *USER_ID*

MIME-TYPE : Could be one of type given PUT CONTENT command

REQUESTED_BY *USER_ID :* Refers to user who execute the command.

Command Definition : Listing by mime-type operation will be done by this command. The requestor has to have listing ability.

Result Information : *List of contents returned from mime-type filtering operation*

Log Information : *USER_FULLNAME* LISTED BY MIME-TYPE

Output

Two seperate output files;

1) Log File

   Logging system activities. Log informations will be written down this file given as above. If a non-authorized user is determined, just write PERMISSION DENIED.

2) Query Result File

   Results of the command list given by input file. Result informations will be written down this file given as above. If a non-authorized user is determined, just write PERMISSION DENIED.

Sample INPUT/OUTPUT Scenario

input.inp

PUT ROLE 1,admin,wrl

PUT ROLE 2,employee,rl

PUT ROLE 3,visitor,r

PUT USER 101,Norah Jones,1

PUT USER 102,Caro Emerald,2

PUT USER 103,Serdar Ortac,3

PUT CONTENT 1001,document1.txt,TXT,{545} REQUESTED_BY 101

PUT CONTENT 1002,document2. pdf, PDF,{1654} REQUESTED_BY 101

PUT CONTENT 1003,document3,PDF,{23545} REQUESTED_BY 102

PUT CONTENT 1004,video1.avi,AVI,{24,600x800,120} REQUESTED_BY 102

PUT CONTENT 1005,image1.jpg,JPG,{1980x1080} REQUESTED_BY 101

PUT CONTENT 1006,image2.png, PNG,{ 1600x1200} REQUESTED_BY 103

GET 1001 REQUESTED_BY 103

GET 1004 REQUESTED_BY 102

GET 1005 REQUESTED_BY 101

GET 1005 REQUESTED_BY 102

LIST BY SIZE REQUESTED_BY 102

LIST BY TYPE PDF REQUESTED_BY 103

LIST BY TYPE PDF REQUESTED_BY 101

Output Files

result.out

Command:PUT ROLE 1,admin,wrl

admin role is added

Command:PUT ROLE 2,employee,rl

employee role is added

Command:PUT ROLE 3,visitor,r

visitor role is added

Command:PUT USER 101,Norah Jones,1

A(n) admin is added:Norah Jones

Command:PUT USER 102,Caro Emerald,2

A(n) employee is added: Caro Emerald

Command:PUT USER 103,Serdar Ortac,3

A(n) visitor is added:Serdar Ortac

Command:PUT CONTENT 1001,document1.txt,TXT,{545} REQUESTED_BY 101

Norah Jones put a document with size of 545

Command:PUT CONTENT 1002,document2. pdf, PDF,{1654} REQUESTED_BY 101

Norah Jones put a document with size of 1654

Command:PUT CONTENT 1003,document3,PDF,{23545} REQUESTED_BY 102

Caro Emerald put a document with size of 23545

Command:PUT CONTENT 1004,video1.avi,AVI,{24,600x800,120} REQUESTED_BY 102

Caro Emerald put a video with size of 4147200000

Command:PUT CONTENT 1005,image1.jpg,JPG,{1980x1080} REQUESTED_BY 101

Norah Jones put an image with size of 6220800

Command:PUT CONTENT 1006,image2.png, PNG,{ 1600x1200} REQUESTED_BY 103

PERMISSION DENIED

Command:GET 1001 REQUESTED_BY 103

document1.txt viewing by Serdar Ortac

Command:GET 1004 REQUESTED_BY 102

video1.avi streaming by Caro Emerald

Command:GET 1005 REQUESTED_BY 101

image1.jpg drawing by Norah Jones

Command:GET 1005 REQUESTED_BY 102

image1.jpg drawing by Caro Emerald

Command:LIST BY SIZE REQUESTED_BY 102

1-video1.avi,4147200000

2-image1.jpg,6220800

3-document3,23545

4-document2.pdf,1654

5-document1.txt,545

Command:LIST BY TYPE PDF REQUESTED_BY 103

PERMISSION DENIED

Command:LIST BY TYPE PDF REQUESTED_BY 101

1-document2.pdf

2-document3

log.out

ROLE ADDED

ROLE ADDED

ROLE ADDED

USER ADDED

USER ADDED

USER ADDED

A DOCUMENT ADDED

A DOCUMENT ADDED

A DOCUMENT ADDED

A VIDEO ADDED

AN IMAGE ADDED

PERMISSION DENIED

A DOCUMENT IS READ

A VIDEO IS READ

AN IMAGE IS READ

AN IMAGE IS READ

Caro Emerald LISTED BY SIZE

PERMISSION DENIED

Norah Jones LISTED BY MIME-TYPE

4. SUBMISSION

<student_id>
<report>
report.pdf
<src>
Main.java
***.java**

5. NOTES AND RESTRICTIONS

- You will use online submission system to submit your experiment. **No other submission method** such as CD, USB, e-mail will be accepted.
- Submission time for deadline is 17:00. Submit system will stay open until 23.59 at deadline**, but any problem you faced after 17:00 will be under your responsibility**. We have no physical access chance during the evening.
- Do not submit any file via e-mail related with this assignment
- Save and don't share all your work until the assignment is graded announced at the end of reclamation period.
- The assignment must be original, **INDIVIDUAL** work. **DUPLICATE** or **VERY SIMILAR ASSIGNMENTS** are both going to be punished rigidly. General discussion of the problem is allowed, but **DO NOT SHARE YOUR DESIGN OR IMPLEMENTATION**.
- You can ask your questions through course's piazza page:
  - https://piazza.com/hacettepe.edu.tr/summer2014/bbm104/home
- You are supposed to be aware of everything discussed in this page
- Don't ask before google it.

6. REFERENCES

1- Head First Java;Kathy Sierra, Bert Bates; O'Reilly Media; 2nd edition (February 9, 2005) ISBN-10: 0596009208
2- http://docs.oracle.com/javase/tutorial/
3- http://www.google.com