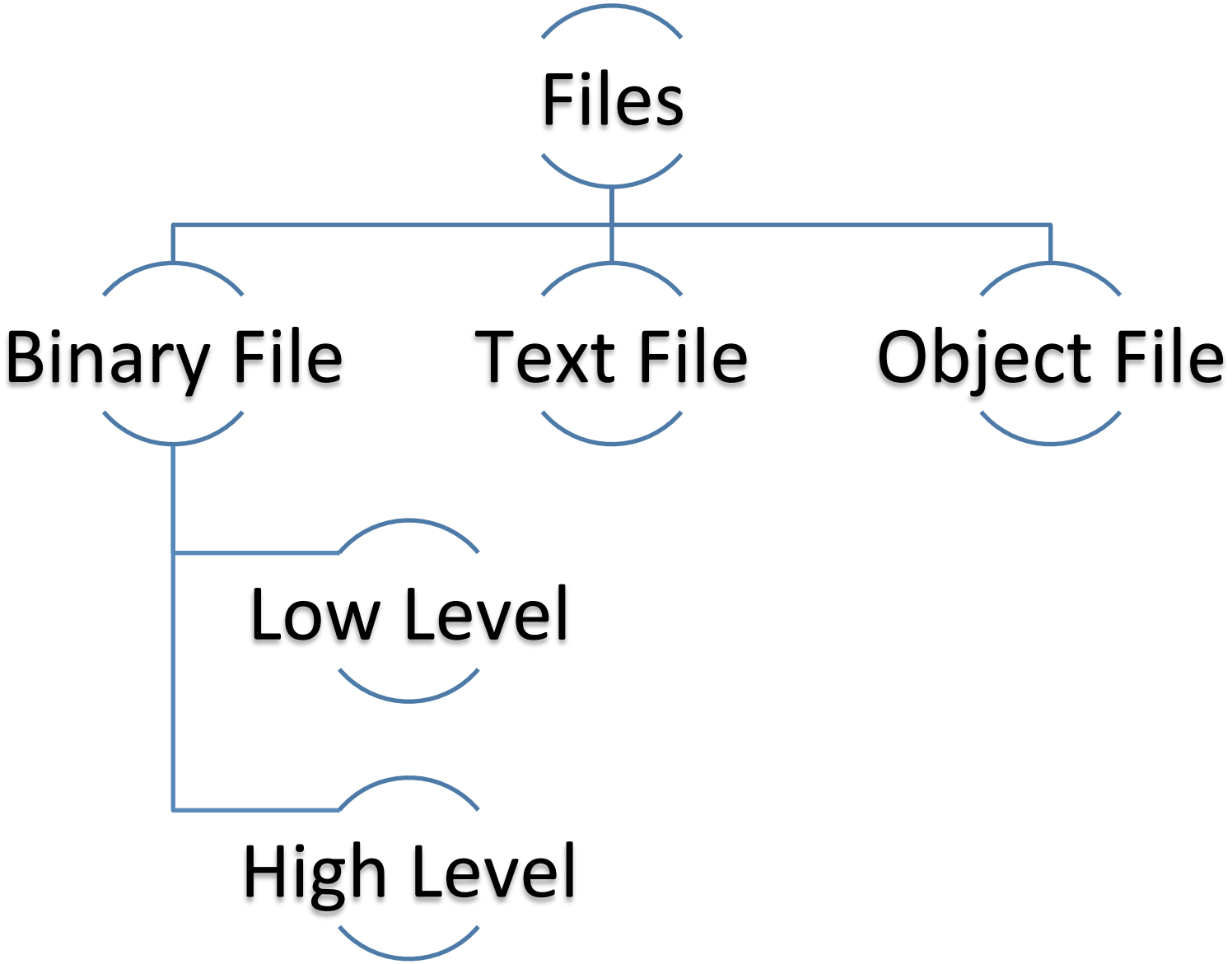
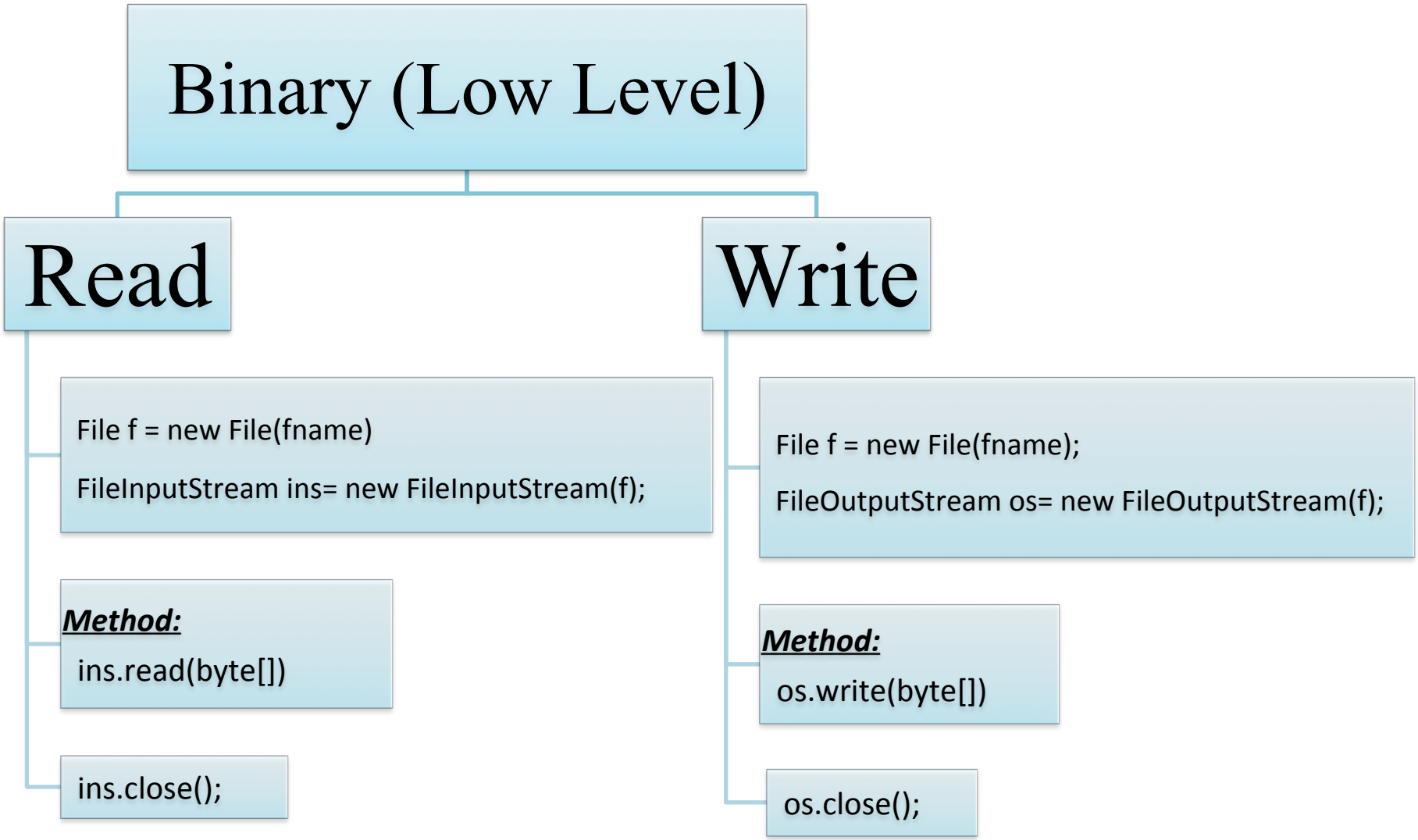
Files Extra Example


Files
Binary File
Text File
Object File
Low Level
High Level


Binary (Low Level)
Read
File f = new File(fname)
FileInputStream ins= new FileInputStream(f);
Method:
ins.read(byte[])
ins.close();
Write
File f = new File(fname);
FileOutputStream os= new FileOutputStream(f);
Method:
os.write(byte[])
os.close();

Binary (High Level)

Read

```
File f = new File(fname)
FileInputStream is= new FileInputStream(f);
DataInputStream DS = new DataInputStream(is);
```

Method:

DS.readInt()

DS.readByte()

etc..

```
DS.close();
```

Write

```
File f = new File(fname);
FileOutputStream os= new FileOutputStream (f);
DataOutputStream Dos = new DataOutputStream(os);
```

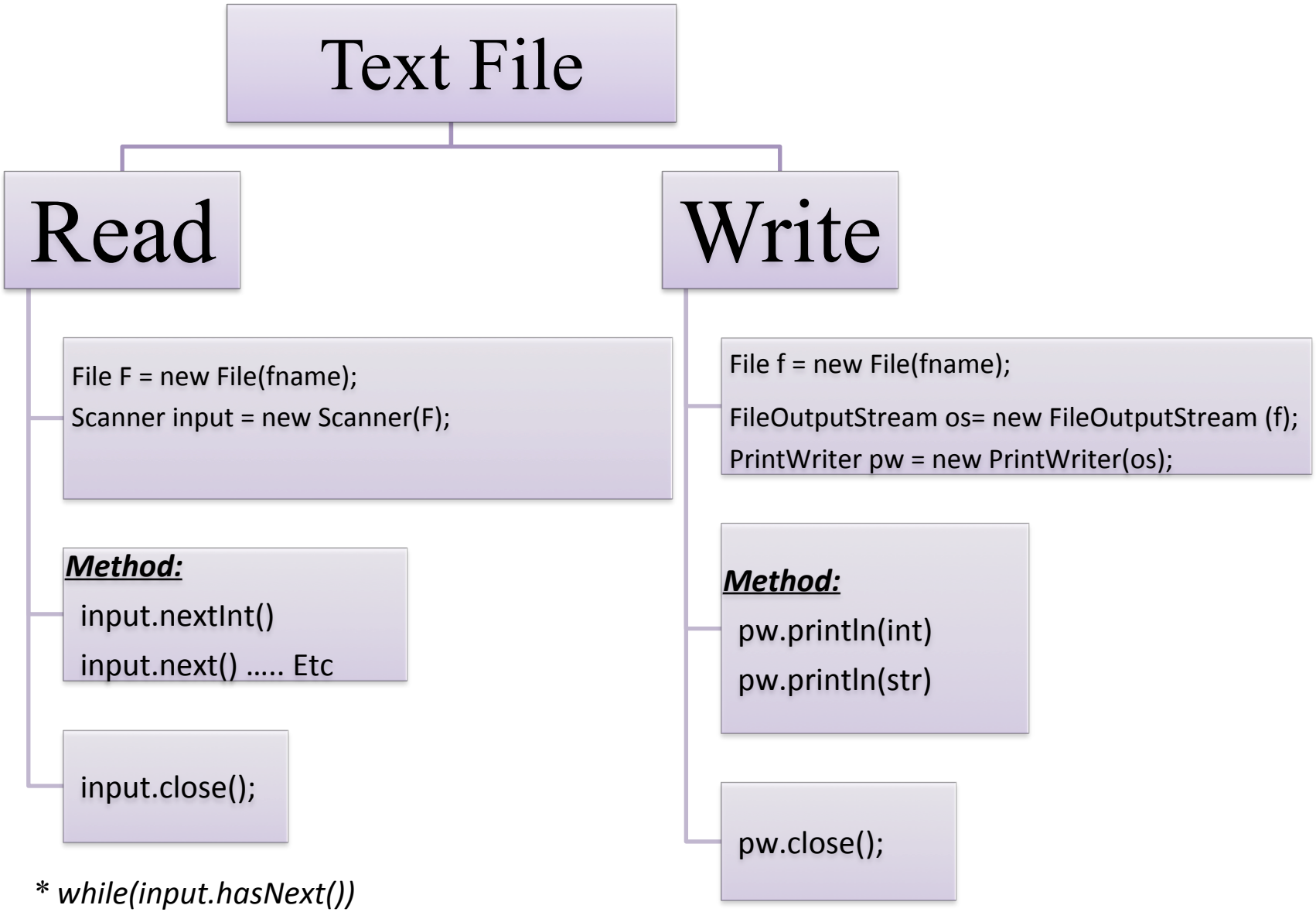
Method:

Dos.writeInt(int)

Dos.writeByte(byte)

etc..

```
Dos.close();
```



* *while(input.hasNext())*

Object File

Read

Exception:
IOException
ClassNotFoundException

File f= new File(Filename)

FileInputStream fileInput= new FileInputStream(f);
ObjectInputStream input= new ObjectInputStream (fileInput);

Method:

(cast) input.readObject();

input.readLine();

input.close();

* *EOFException*

Write

Exception:
IOException

File f= new File(Filename)

FileOutputStream outFile= new FileOutputStream(f);

ObjectOutputStream out= new ObjectOutputStream (outFile);

Method:

out.writeObject(obj);

out.writeInt(1);

out.writeBytes("Str");

out.close();

Dealing with object files

Read (input) & Write (Output)

```
import java.io.*;
public class Person
implements Serializable
{
String name;
int age;
public Person(String n,int
a){
      name=n;
      age=a;
}
public String toString()
{
  return name+" "+ age;
} }
```

```
import java.io.*;
public class ObjTest{
public static void main (String [] args)
throws IOException {
FileOutputStream outFile= new
FileOutputStream(new File("sample1.data"));

ObjectOutputStream out= new
ObjectOutputStream (outFile);

Person p = new Person("Ali", 20);
out.writeObject(p);
out.close();
}}
```

```
import java.io.*;
public class ObjRead {
public static void main (String [] args)throws IOException,
ClassNotFoundException {
FileInputStream fileInput= new FileInputStream(new File
("sample1.data"));
ObjectInputStream input= new ObjectInputStream (fileInput);
Person p=(Person)input.readObject();
System.out.println(p);
input.close();
}}
```

The use of EOF Exception

Q1: Assume the following implementation of classes: Game, tabletopGame, and videoGame:

```
import java.io.*;
public class Game implements Serializable {
private String name;
private double price;
public Game(String n, double p) {name=n;
 price=p;}
// ..
}
public class TabletopGame extends Game {
//...
}
public class VideoGame extends Game{
//...
}
```

Given an object file "*gamesInventory.dat*" that stores all the games currently displayed in a local shop.

Write a program **that reads the content of the file "*gamesInventory.dat*", counts the number of games per type**, and **calculates their average price**. Then prints this information on the screen.

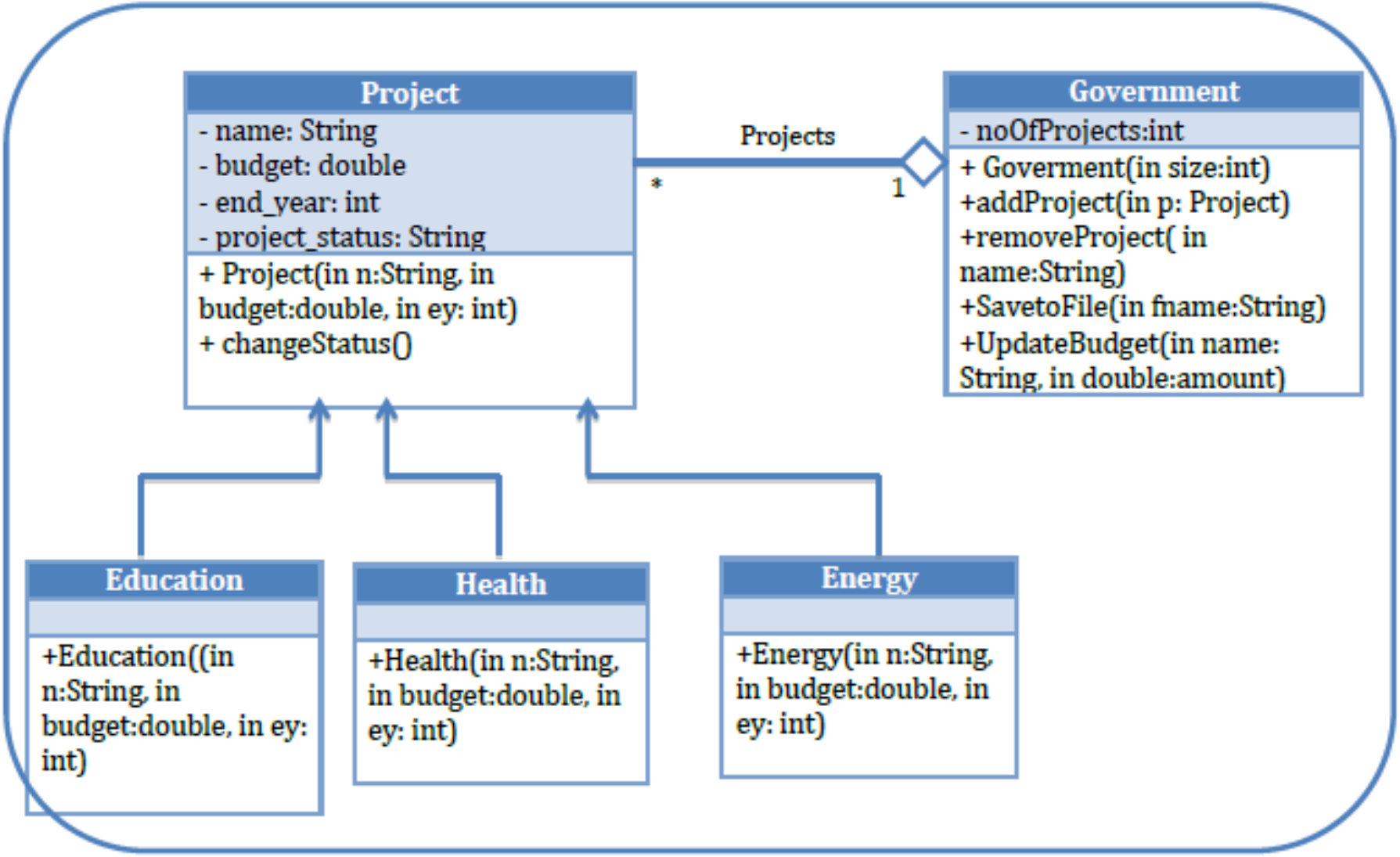
```
import java.io.*;
public class T10Q1 {
    public static void main(String []args)throws IOException {
      File F = new File("gamesInventory.dat");
      FileInputStream Fin= new FileInputStream(F);
      ObjectInputStream Oin = new ObjectInputStream(Fin);
      int Tcount=0, Vcount=0; double Tsum=0, Vsum=0;
      Game obj=null;
     try {
         while(true) {
            try {
               obj= (Game)(Oin.readObject());
               if (obj instanceof TabletopGame) {
                  Tcount++;
                  Tsum+=obj.getPrice(); }

               else if (obj instanceof VideoGame) {
                  Vcount++;
                  Vsum+=obj.getPrice();} }

            catch(ClassNotFoundException e) {System.out.println(e);}
         }
      }
      catch (EOFException e) {Oin.close();}
          System.out.println("The store has "+Tcount+" tabletop games,
with average price= "+(Tsum/Tcount));
            System.out.println("The store has "+Vcount+" video games,
with average price= "+(Vsum/Vcount));
```

Extra Example

Q1: Given the following UML and the corresponding class descriptions below:



1- The Java code for all classes have been provided. Please complete the missing methods only.

2- Create any customized Exception classes you need

Class project

Project(String n, double budget, int ey):

- The constructor should check that the ***budget*** is in the range 1 to 10 million and throw a ***BUDGETOutOfRangeException*** otherwise.
- The constructor should check also the end_year, it should be an integer of four digits that starts with 20 such as 2015 and throw a suitable exception otherwise.

```
public class Project {
   private String name;
   private double budget;
   private String pro_status;
   private int end_year;

   public Project(String n, double b, int ey) throws
BudgetOutOfReangeException {
      if(b<1 || b >10)
         throw new BudgetOutOfReangeException("Out of range");
      if (!(ey/100 == 20 && ey/2000 ==1))
         throw new IllegalArgumentException("The year format is wrong");
      name = n;
      budget=b;
      end_year =ey;
      set_status();
   }
   public void set_status(){
      if (end_year == 2016)
         pro_status = "ABOUT TO FINISH”;
      else if (end_year > 2016)
         pro_status = "UNDER PROGRESS";
      else if (end_year < 2016)
         pro_status = "COMPLETE”;
   } }
```

Class government

saveToFile(String fname): This method writes the information of all projects in the received text file ordered by project type.

```
import java.io.*;
public class Government {
   private int noOfProjects;
   private Project[] projects;
   public Government(int size) {
      projects = new Project[size];
   }
   public void addProject (Project s) {
      projects[noOfProjects]= s;
      noOfProjects++;
   }
   public void RemoveProject(int loc) {
      if (projects[loc].getStatus().equals("COMPLETE")) {
         for (int i=loc ; i<noOfProjects; i++)
            projects[i]=projects[i+1];
         noOfProjects--;
      }}
```

```
 public void saveToFile(String fname) throws IOException {
      File f = new File(fname);
      FileOutputStream os= new FileOutputStream (f);
      PrintWriter pw = new PrintWriter(os);

      pw.println("Education:");
      for (int i=0; i<noOfProjects; i++)
         if (projects[i] instanceof Education)
            pw.println(projects[i].getName() + "   " +
projects[i].getbudget()+ "   "+ projects[i].getStatus());

      pw.println("Health:");
     for (int i=0; i<noOfProjects ; i++)
         if (projects[i] instanceof Health)
            pw.println(projects[i].getName() + "   " +
projects[i].getbudget()+ "   "+ projects[i].getStatus());

      pw.println("Energy:");
      for (int i=0; i<noOfProjects ; i++)
         if (projects[i] instanceof Energy)
            pw.println(projects[i].getName() + "   " +
projects[i].getbudget()+ "   "+ projects[i].getStatus());

      pw.close();
}
```

Class government

updateBudget(String name, double amount): This method updates the budget of the project with the name '***name'*** by the received ***amount***.

- If the project status is UNDER PROGRESS the ***BUDGET*** increased by the amount.
- If the project status is COMPLETE the ***BUDGE*** decreased by the amount.
- If the new ***BUDGET*** is less than or equals 0 the method should throw a ***lowBUDGETException***. Otherwise the method should update the budget with the new one

This method also handles the ***lowBUDGETException*** by printing a suitable message.

```
public void updateBudget(String name, double amount) {
  double new_budget = 0;
      for (int i=0; i<noOfProjects ; i++)
         try{
            if(projects[i].getName().equals(name)) {
               if((projects[i].getStatus().equals("UNDER
PROGRESS"))||(projects[i].getStatus().equals("ABOUT TO FINISH") )
               new_budget = (projects[i].getbudget())+amount;

               else if(projects[i].getStatus().equals("COMPLETE"))
               new_budget = (projects[i].getbudget())-amount;

               if(new_budget <=0)
                  throw new LowBudgetException("LOW BUDGET!!");
               else
               projects[i].setbudget(new_budget);
            }
         }
         catch (LowBudgetException e) {
            System.out.println(e.getMessage()); }
   }
```

Given Code for Class Education, Energy and Health

```
public class Education extends Project {
public Education(String n, double b, int ey) throws BudgetOutOfReangeException
{
      super(n,b,ey);
}}

public class Energy extends Project {
   public Energy(String n, double b, int ey) throws BudgetOutOfReangeException


{
      super(n,b,ey);
}}

public class Health extends Project {
   public Health(String n, double b, int ey) throws BudgetOutOfReangeException
      super(n,b,ey);
}}
public class LowBudgetException extends Exception {
   public LowBudgetException(String msg) {
      super (msg);
   }
}
```

Test class

- **Write a Test class that performs the following:**
  - Create a Government object of your choice.
  - Display the following menu to the user:
    - Add a project. (Ask the user to enter the project's type (Education, Health, Energy), name and budget)
    - Remove a project. (Ask the user to enter the name of the project he wants to delete)
    - Save project info to a text file. (Ask the user to enter the file name)
    - Update the budget of a project. (Ask the user to enter the name of the project and the budget amount)
    - Exit.
- **Note:** The main method should handle all exceptions that may occur in the program (except for ***lowBUDGETException***), in a way that won't affect the flow of the program.

```
import java.util.*; import java.io.*;
public class test {
static Scanner read = new Scanner(System.in);
 public static void main(String[] args) {
  Government gov = new Government(100);
  int choice;
   do{
    System.out.println("Choose one of the following options:");
    System.out.println("1- Add a project.");
    System.out.println("2- Remove a project.");
    System.out.println("3- Save projects info to a file.");
    System.out.println("4- Update the budget of a project.");
    System.out.println("5- Exit.");
    choice= read.nextInt();

     switch(choice) {
      case 1:
       System.out.println("Please enter the type of the
project");
       String type = read.next();
       System.out.println("Please enter the project's name");
       String name = read.next();
```

```
System.out.println("Please enter the project's budget and the end
year");
        boolean done = false;
        int ey = 2000; double budget =0;
          while(!done){
             try{
               budget = read.nextDouble();
               ey = read.nextInt();
               Project p;
                if(type.equals("Education"))
                   p = new Education(name, budget, ey);
                else if (type.equals("Health"))
                   p = new Health(name, budget, ey);
                else
                   p = new Energy(name, budget, ey);
                gov.addProject(p);
               done = true;}
             catch(InputMismatchException e) { read.next();
                System.out.println("please enter a double");}
             catch(IllegalArgumentException e)
{ System.out.println(e.getMessage()+ " please enter again"); }
             catch (BudgetOutOfReangeException e)
{ System.out.println(e.getMessage()+ " please enter again");}
               }
break;
```

```
 case 2:
                System.out.println("please enter the location of
the project you want to remove");
                int loc = 0;
                done= false;
                while(!done)
                   try{
                      loc = read.nextInt();
                      done=true;
                   }
                   catch(InputMismatchException e)
                   {
                      read.next();
                      System.out.println("please enter an int");
                   }
                try{ // can be embedded in one try
                   gov.RemoveProject(loc);
                }
                catch (ArrayIndexOutOfBoundsException e)
                {
                   System.out.println("No such location");
                }
               break;
```

```
case 3:
  System.out.println("please enter the file name");

    try
     {
        gov.saveToFile(read.next());
     }
     catch (IOException e)
     {
          System.out.println("Problem in file proccessing");
     }
     break;
```

```
 case 4:
                System.out.println("Please enter the project's
name");
                String name1 = read.next();
                System.out.println("Please enter the amount");
                done = false;
                double amount =0;
                while(!done){
                   try{
                      amount = read.nextDouble();
                      done = true;}
                   catch(InputMismatchException e)
                   {
                      read.next();
                      System.out.println("please enter a double");
                   }
                }
                gov.updateBudget(name1, amount);
                break;
          }//end switch

       }while(choice !=5);
   }}
```