**SHOP**

String shopName;

ArrayList<Product> items;

**Exceptions**

Illegal Age,Name,age,sex

**Author**

**private** String id;

**private** String fName;

**private** String lName;

**private** **double** age;

**private** String sex;

**Product**

**public** **static** **int** *count*=0;

**protected** **int** Sn;

**protected** String Name;

**protected** **int** publishDate;

**protected** **int** price;

**Book**

**protected** ArrayList<Author> Authors = **new** ArrayList<Author>();

**protected** **int** pagesNum;

**protected** genre genre;

**Gameboard**

**protected** **int** soldiersNum;

**protected** String incDices;

**protected** String incCards;

**Digital**

**protected** **int** size;

**protected** String downable;

**Audiobook**

**protected** String tapeName;

**protected** **int** audioTime;

**VideoGame**

**protected** GameCompanie GameCompanie;

**protected** genre genre;

**protected** String singlePalyerOnly;

**Interfaces**

**public** **interface** Multiplayerable

**Exceptions**

**public** **class** IllegalArgument **extends** Exception

**Enums**

**public** **enum** GameCompanie

**Author**

**import** java.io.Serializable;

**public** **class** Author **implements** Serializable

{

**private** String id;

**private** String fName;

**private** String lName;

**private** **double** age;

**private** String sex;

//sets

**public** **void** setId(String id) **throws** IllegalID

{

//תנאי שווה ל9 תווים ומכיל מספרים בלבד

**char**[] chars = id.toCharArray();

**if**(chars.length == 9) {

**for** (**char** c : chars) {

**if** (!Character.*isDigit*(c))

**throw** **new** IllegalID(id);}

**this**.id=id;}

}

}

**Exeption**

**public** **void** setAge(**double** age) **throws** IllegalAge

{

//תנאי מספר לא שלם גדול מ0

**if**(age>0 )

**this**.age = age;

**else**

**throw** **new** IllegalAge(age);

}

//constructor

**public** Author(String id, String fName,

String lName, **double** age,String sex)

**throws** IllegalName,IllegalGender,IllegalAge,IllegalID

{

setId(id);

setfName(fName) ;

setlName(lName) ;

setAge(age);

setSex(sex);

}

**public** **void** setfName(String fName) **throws** IllegalName

// תנאי גדול משלוש תווים ומכיל אותיות בלבד

{

**char**[] chars = fName.toCharArray();

**if**(chars.length>=3) {

**for** (**char** c : chars) {

**if** (!Character.*isLetter*(c))

**throw** **new** IllegalName(fName);}

**this**.fName=fName;}

**public** **void** setlName(String lName) **throws** IllegalName

{

// תנאי גדול משלוש תווים ומכיל אותיות בלבד

**char**[] chars = lName.toCharArray();

**if**(chars.length>=3) {

**for** (**char** c : chars) {

**if** (!Character.*isLetter*(c))

**throw** **new** IllegalName(lName);}

**this**.lName=lName;}

}

**public** String getId() {

**return** id;

}

**public** String getfName() {

**return** fName;

}

**public** String getlName() {

**return** lName;

}

**public** **double** getAge() {

**return** age;

}

**public** String getSex() {

**return** sex;

**public** **boolean** equals(Object obj)

{

//מקבלת אובייקט וומשווה לסופר

**return** (obj **instanceof** Author && **this**.id.equals(((Author)obj).id));

} }

**public** **class** IllegalAge **extends** Exception

{

**public** IllegalAge(**double** age)

{

**super**(age+" Illegal Age ");

}

}

**public** String toString()

{

**return** "Author"+"[ID : "+id+"|

First name : "+fName+" "

+"|Last name : "+lName+"| Age : "+age+"| Gender : "+sex+"]";

}

**Product**

//sets

**abstract** **public** **void** setPublishDate(**int** publishDate) **throws** IllegalArgument;

**abstract** **public** **void** setPrice(**int** price) **throws** IllegalArgument;

**public** **void** setName(String name) **throws** IllegalArgument

{String msg = "year must be over 3 letters";

**if** (name.length()>=3)

Name = name;

**else**

**throw** **new** IllegalArgument(msg);

}

@Override

**public** String toString() {

**return** publishDate+","+ price +","+ Sn +","+ Name+","+ ; }

@Override

**public** **boolean** equals(Object obj) {

**return**

**this**.getClass().getName().equals(obj.getClass().getName())

&& **this**.Sn == ((Product)obj).Sn;

}

public abstract class Product implements Serializable{

public static int *count*=0;

protected int Sn;

protected String Name;

protected int publishDate;

protected int price;

public Product(String name, int publishDate, int price)

throws IllegalArgument

{

*count*++;

this.Sn=*count*;

setName(name);

setPublishDate(publishDate);

setPrice(price);

}

//gets

public String getName() {

return Name;

}

public int getPublishDate() {

return publishDate;

}

public int getPrice() {

return price;

}

**public****int** getSn() {

**return** Sn;

}

**GameBoard**

@Override

**public** String toString() {

**return** **super**.toString()+"," + soldiersNum + "," + incDices + "," + incCards ;}

@Override

**public** **boolean** equals(Object obj) {

**return** **super**.equals(obj);}

**public** **boolean** Multiplayerable() {

**if**(**this**.soldiersNum>1)

**return** **true**;

**else**

**return** **false**;}

**public** **class** GameBoard **extends** Product **implements** Multiplayerable

{

**protected** **int** soldiersNum;

**protected** String incDices;

**protected** String incCards;

**public** GameBoard(String name, **int** publishDate, **int** price,**int** soldiersNum, String incDices, String incCards)

**throws** IllegalArgument {

**super**(name, publishDate, price);

setSoldiersNum(soldiersNum);

setIncDices(incDices);

setIncCards(incCards);

}

**public** **int** getSoldiersNum() {

**return** soldiersNum;}

**public** String getIncDices() {

**return** incDices;

**public** String getIncCards() {

**return** incCards; }

@Override

**public** **void** setPublishDate(**int** publishDate) **throws** IllegalArgument {

String msg = "year must be over 1500";

**if** (publishDate>1950)

**this**.publishDate=publishDate;

**else**

**throw** **new** IllegalArgument(msg); }

@Override

**public** **void** setPrice(**int** price) **throws** IllegalArgument{

String msg = "price must be btween 30 and 120";

**if**(price>30 && price <120)

**this**.price=price;

**else**

**throw** **new** IllegalArgument(msg); }

**public** **void** setSoldiersNum(**int** soldiersNum) **throws** IllegalArgument{

String msg = "Answer cannot be negetive";

**if**(soldiersNum>=0)

**this**.soldiersNum = soldiersNum;

**else**

**throw** **new** IllegalArgument(msg); }

**public** **void** setIncDices(String incDices)**throws** IllegalArgument{

String msg = " Answer must be yes or no";

**if**(incDices.equalsIgnoreCase("yes")

||incDices.equalsIgnoreCase("no"))

**this**.incDices = incDices;

**else**

**throw** **new** IllegalArgument(msg);

}

**public** **void** setIncCards(String incCards)**throws** IllegalArgument {

String msg = " Answer must be yes or no";

**if**(incCards.equalsIgnoreCase("yes")

||incCards.equalsIgnoreCase("no"))

**this**.incCards = incCards;

**else**

**throw** **new** IllegalArgument(msg);

}

**Book**

import java.util.ArrayList;

public class Book extends Product implements Comparable<Book>

{

protected ArrayList<Author> Authors

= new ArrayList<Author>();

protected int pagesNum;

protected genre genre;

public Book( String name, int publishDate, int price,

ArrayList<Author> authors, int pagesNum,genre genre) throws IllegalArgument {

{super( name, publishDate, price);

setAuthors(authors);

setpagesNum(pagesNum);

setGenre(genre);

public ArrayList<Author> getAuthors() {

return Authors;

}

public int getPagesNum() {

return pagesNum;

}

public genre getGenre() {

return genre;

}

**public** **void** setAuthors(ArrayList<Author> authors) {

Authors = authors;}

**public** **void** setGenre(genre genre) {

**this**.genre = genre;}

**public** **void** setpagesNum(**int** pagesNum) **throws** IllegalArgument {

String msg = "Number of page";

**if**(pagesNum>0)

**this**.pagesNum = pagesNum;

**else**

**throw** **new** IllegalArgument(msg);}

@Override

public int compareTo(Student s) {

if (name.equals(s.name))

return 0;

return name.compareTo(s.name)); }

@Override

**public** **void** setPublishDate(**int** publishDate) **throws** IllegalArgument {

String msg = "year must be over 1918";

**if** (publishDate>1918)

**this**.publishDate=publishDate;

**else**

**throw** **new** IllegalArgument(msg);}

@Override

**public** **void** setPrice(**int** price) **throws** IllegalArgument{

String msg = "price is btween 20 and 100";

**if**(price>20 && price <110)

**this**.price=price;

**else**

**throw** **new** IllegalArgument(msg);}

@Override

**public** String toString() {

String author=" ";

**for** (Author a : Authors) {

author +=a.getfName()+" "+a.getlName()+" ";}

**return** **super**.toString()+","+author+ "," + pagesNum +","+ genre;}

@Override

**public** **boolean** equals(Object obj) {

**return** **super**.equals(obj);}

@Override

**public** **int** compareTo(Book b){

**if** (**this**.pagesNum < b.pagesNum) //this comes 1st

**return** -1;

**else** **if**(**this**.pagesNum > b.pagesNum)//b comes 1st

**return** 1;

**else** **return** 0;//this and b are equal

**Digital**

**abstract** **public** **void** setSize(**int** size) **throws** IllegalArgument;

**public** **void** setAvailable(String available) **throws** IllegalArgument {

String msg = " Answer must be yes or no";

**if**(available.equalsIgnoreCase("yes")

||available.equalsIgnoreCase("no"))

**this**.available = available;

**else**

**throw** **new** IllegalArgument(msg);}

@Override

**public** String toString() {

**return** **super**.toString()+","+size+"," + available ;}

@Override

**public** **boolean** equals(Object obj)

{

**return** **super**.equals(obj);}

**package** course.projects.products;

**public** **abstract** **class** Digital **extends** Product {

**protected** **int** size;

**protected** String available ;

**public** Digital(String name, **int** publishDate, **int** price,

**int** size, String available) **throws** IllegalArgument{

**super**(name, publishDate, price);

setSize(size);

setAvailable(available);

}

**public** **int** getSize() {

**return** size;

}

**public** String getAvailable() {

**return** available;

}

**VideoGame**

@Override

**public** **void** setPublishDate(**int** publishDate) **throws** IllegalArgument {

String msg = "year must be over 2000";

**if** (publishDate>2000)

**this**.publishDate=publishDate;

**else**

**throw** **new** IllegalArgument(msg);}

@Override

**public** **void** setPrice(**int** price) **throws** IllegalArgument{

String msg = "price is btween 0 and 500";

**if**(price>=0 && price <500)

**this**.price=price;

**else**

**throw** **new** IllegalArgument(msg);}

**public** **void** setSize(**int** size )**throws** IllegalArgument {

String msg = "Size must be over 0 and under 60000";

**if**(size>0 && size<60000)

**this**.size = size;

**else**

**throw** **new** IllegalArgument(msg);}

**public** **void** setSinglepalyer(String singlePalyerOnly) **throws** IllegalArgument {

String msg = " Answer must be yes or no";

**if**(singlePalyerOnly.equalsIgnoreCase("yes")

||singlePalyerOnly.equalsIgnoreCase("no"))

**this**.singlePalyerOnly = singlePalyerOnly;

**else**

**throw** **new** IllegalArgument(msg);}

**public** **class** VideoGame **extends** Digital **implements** Multiplayerable{

**protected** GameCompanie GameCompanie;

**protected** genre genre;

**protected** String singlePalyerOnly;

**public** VideoGame(String name, **int** publishDate, **int** price,**int** size, String available, GameCompanie ,genre genre, String singlePalyerOnly) **throws** IllegalArgument {

**super**(name, publishDate, price,size,available);

**this**.GameCompanie=GameCompanie;

**this**.genre=genre;

setSinglepalyer(singlePalyerOnly);}

**public** GameCompanie getGameCompanie() {

**return** GameCompanie;}

**public** genre getGenre() {

**return** genre;}

**public** String getSinglepalyer() {

**return** singlePalyerOnly;}

import java.util.ArrayList;

public class Book extends Product implements Comparable<Book>

{

protected ArrayList<Author> Authors

= new ArrayList<Author>();

protected int pagesNum;

protected genre genre;

public Book( String name, int publishDate, int price,

ArrayList<Author> authors, int pagesNum,genre genre) throws IllegalArgument {

super( name, publishDate, price);

setAuthors(authors);

setpagesNum(pagesNum);

setGenre(genre);

public ArrayList<Author> getAuthors() {

return Authors;

}

public int getPagesNum() {

return pagesNum;

}

public genre getGenre() {

return genre;

}

@Override

**public** String toString() {

**return** **super**.toString( )+"," + GameCompanie

+ ", " + genre + "," + singlePalyerOnly ;}

@Override

**public** **boolean** equals(Object obj) {

**return** **super**.equals(obj);}

**public** **boolean** Multiplayerable() {

**if**(**this**.singlePalyerOnly.equalsIgnoreCase("yes"))

**return** **false**;

**else**

**return** **true**;}

**AudioBook**

@Override

**public** **void** setPublishDate(**int** publishDate) **throws** IllegalArgument {

String msg = "year must be over 1918";

**if** (publishDate>2010)

**this**.publishDate=publishDate;

**else**

**throw** **new** IllegalArgument(msg);}

@Override

**public** **void** setPrice(**int** price) **throws** IllegalArgument{

String msg = "price is btween 50 and 150";

**if**(price>50 && price <150)

**this**.price=price;

**else**

**throw** **new** IllegalArgument(msg);

@Override

**public** **boolean** equals(Object obj) {

**return** **super**.equals(obj);}

@Override

**public** String toString() {

**return** **super**.toString()+"," + size+","+ tapeName +"," + audioTime+","available;

**public** **void** setSize(**int** size )**throws** IllegalArgument {

String msg = "size must be over 1000 and under 2000";

**if**(size>1000 && size<2000)

**this**.size = size;

**else**

**throw** **new** IllegalArgument(msg);}

**public** **void** setTapeName(String tapeName) **throws** IllegalArgument {

String msg = "tape name must be letters only";

**char**[] chars = tapeName.toCharArray();

**for** (**char** c : chars)

{

**if** (Character.*isDigit*(c))

**throw** **new** IllegalArgument(msg);

}

**this**.tapeName = tapeName;}

**public** **class** AudioBook **extends** Digital

{

**protected** String tapeName;

**protected** **int** audioTime;

**public** AudioBook(String name, **int** publishDate, **int** price

,**int** size,String tapeName, **int** audioTime, String available)**throws** IllegalArgument {

**super**(name, publishDate, price,size,available);

setTapeName(tapeName);

setAudioTime(audioTime);}

**public** **int** getSize() {

**return** size;}

**public** String getTapeName() {

**return** tapeName;}

**public** **int** getAudioTime() {

**return** audioTime;}

**public** String isAvailable() {

**return** available;}

public enum genre

{

*Horror*,*Action*,*Adventure*,*Comedy*,

}

**Enum**

**Interface**

**Exeption**

**public** **class** IllegalArgument **extends** Exception {

**public** IllegalArgument(String msg) {**super**(msg);} {

public interface Multiplayerable

{

boolean Multiplayerable();

}

**Shop**

@Override

**public** String toString(){

String str = "";

**for**(Product p : items)

str += "\n"+p;

**return** str; }

**import** java.io.\*;

**import** java.util.ArrayList;

**import** java.util.Collections;

**public** **class** Shop **implements** Serializable{

**private** String shopName;

**private** ArrayList<Product> items;

**public** Shop(String shopName)**throws** IllegalArgument{

setShopName(shopName);

**this**.items = **new**ArrayList<Product>();}

**public** String getShopName() {

**return** shopName;}

**public** ArrayList<Product> getItems() {

**return** items;}

**public** **void** addProduct(Product p)

{

**if**(!items.contains(p))

items.add(p);

**else**

System.***out***.println("item already in store");}

**public** **void** removeProduct(Product p){

**if**(items.contains(p))

items.remove(p);

**else**

System.***out***.println("item isnt in the store");}

**public** String findTheNewestProducts(){

**int** min = 2015;

String newp= "Newest Products :"+"\n";

**for**(Product p : items)

{

**if** (p.publishDate>min)

newp += p+"\n";

}

**return** newp;}

**public** String infdTheCheapestProduct(){

String cheap = "Cheapest Product : "+"\n" ;

**int** min = items.get(0).price;

**for**(Product p : items)

{

**if** (p.price<min)

min=p.price;

}

**for**(Product p : items)

{

**if**(p.price==min)

cheap += p+"\n";

}

**return** cheap;}

**public** String searchByNumOfPagesUnsorted(**int** pages){

String books = "Number Of Pages above "+pages+": "+"\n";

**for**(Product p : items)

{

**if** (p.getClass().getName().equals("Book"))

**if**(((Book) p).pagesNum>pages)

books+=p.toString()+"\n";

}

**return** books;}

===========================================================

**public** String searchByNumOfPages(**int** pages){

ArrayList<Book> Books = **new** ArrayList<Book>();

**for** (Product p : items)

{

**if**(p **instanceof** Book && ((Book)p).pagesNum>pages)

Books.add(((Book)p));

}

Collections.*sort*(Books);

String msg = "Number Of Pages above "+pages+": "+"\n";

**for** (Product b : Books)

msg += b.toString()+"\n";

**return** msg;}

**public** **int** AudioBooksgetNumO(){

**int** count = 0;

**for**(Product p : items)

{

**if** (p **instanceof** AudioBook )

count++;}

**return** count;}

**public** **int** getTotalMegabytes(){

**int** sum = 0;

**for**(Product p : items)

**if**(p.getClass().getSimpleName().equals("AudioBook"))

sum += ((AudioBook)p).size;

**for**(Product p : items)

**if** (p.getClass().getSimpleName().equals("VideoGame"))

sum += ((VideoGame)p).size;

**return** sum; }

===================================================

**public** **int** getTotalMegabytes2(){

**int** sum = 0;

**for**(Product p : items)

**if**(p **instanceof** Digital){

sum += ((Digital)p).getSize();}

**return** sum;}

**Hop**

**Main**

**main**

**import** java.io.IOException;

**import** java.util.ArrayList;

**public** **class** Main {

{

Author a = **new** Author("123456789", "nahman", "bialik", 24, "Male");

Author a2 = **new** Author("123456784", "smadar", "shir", 35, "Fmale");

ArrayList<Author> Authors1 =**new** ArrayList<Author>();

Authors1.add(a);

ArrayList<Author> Authors2 =**new** ArrayList<Author>();

Authors2.add(a);

Authors2.add(a2);

Book b1 = **new** Book("Don Quixote", 1999, 100, Authors1 , 555, genre.***Horror***);

Book b2 = **new** Book("Moby Dick", 2015, 80, Authors1, 333,genre.***Action***);

Book b3 = **new** Book("Kama Sutra", 1995, 80, Authors2, 444,genre.***Comedy***);

AudioBook ad1 = **new** AudioBook("Odyssey", 2016, 100, 1500, "sony", 60, "yes");

AudioBook ad2 = **new** AudioBook("harry potter", 2011, 100, 1500, "sony", 60, "yes");

GameBoard g1 =**new** GameBoard("Photosynthesis",2016, 119, 5, "yes", "yes");

GameBoard g2 = **new** GameBoard("Nemo's War", 2011, 109, 0, "no", "no");

VideoGame vg1 = **new** VideoGame("Starcraft", 2018, 300, 40000, "yes",

GameCompanie.***Blizzard***, genre.***Action***, "yes");

VideoGame vg2 = **new** VideoGame("Warcraft", 2011, 400, 50000, "yes",

GameCompanie.***EA***, genre.***Comedy***, "yes");

VideoGame vg3 = **new** VideoGame("GTA", 2011, 400, 10000, "yes",

GameCompanie.***ROCKSTAR***, genre.***Comedy***, "yes");

Shop sh = **new** Shop("shop");

sh.addProduct(b1);sh.addProduct(b2);sh.addProduct(b3);

sh.addProduct(ad1);sh.addProduct(ad2);

sh.addProduct(g1);sh.addProduct(g2);

sh.addProduct(vg1);sh.addProduct(vg2);sh.addProduct(vg3);

sh.removeProduct(ad1);

System.***out***.println(sh.infdTheCheapestProduct());

System.***out***.println(sh.findTheNewestProducts());

System.***out***.print("Number of Audio books available : ");

System.***out***.println(sh.AudioBooksgetNumO());;

System.***out***.println();

System.***out***.println(sh.searchByNumOfPages(300));

System.***out***.print("Total Megabytes : ");

System.***out***.println(sh.getTotalMegabytes());

System.***out***.println();

System.***out***.print(g1.getName()+" is Multiplayerable : ");

System.***out***.println(g1.Multiplayerable());

System.***out***.println();

System.***out***.print(vg1.getName()+" is Multiplayerable : ");

System.***out***.println(vg1.Multiplayerable());

sh.saveData();

Shop sh2 = sh.loadData();

System.***out***.println(sh2);

sh.saveToCSV(); }

**catch** (ClassNotFoundException e){

System.***out***.println(e);

} **catch** (IOException e) {

System.***out***.println(e);

} **catch** (IllegalArgument e) {

System.***out***.println(e);

} **catch** (IllegalName e) {

System.***out***.println(e);

} **catch** (IllegalGender e){

System.***out***.println(e);

} **catch** (IllegalAge e) {

System.***out***.println(e);

} **catch** (IllegalID e) {

System.***out***.println(e);

}

}

}