**CS1083**

**Assignment #2**

**Daniyal Khan**

**3765942**

**Purchasable Interface:**

*public* *interface* Purchasable {

*public* String *getTitle*();

*public* double *getSellingPrice*();

}

**Item.java:**

*public* *abstract* *class* Item *implements* Purchasable, Comparable<Item> {

*private* String title;

*private* double initialPrice;

*public* *Item*(String title, double initialPrice) {

this.*title* = title;

this.*initialPrice* = initialPrice;

}

*public* String *getTitle*() {

*return* title;

}

*public* double *getInitialPrice*() {

*return* initialPrice;

}

*public* String *toString*() {

*return* title;

}

*public* int *compareTo*(Item other) {

int titleComparison = this.*title*.*compareTo*(other.*getTitle*()); // *calculate the difference in the titles aplhabetically*

*if* (titleComparison < 0) { // *we only want to return -1, 1 or 0*

*return* -1;

} *else* *if* (titleComparison > 0) {

*return* 1;

}

// *if title were aplhabetically; compare the prices*

*if* (this.*getSellingPrice*() < other.*getSellingPrice*()) {

*return* -1;

} *else* *if* (this.*getSellingPrice*() > other.*getSellingPrice*()) {

*return* 1;

} *else* { // *if prices and titles were equal return 0*

*return* 0;

}

}

}

**AudioItems.java:**

import *java*.*text*.*NumberFormat*;

*public* *abstract* *class* AudioItems *extends* Item {

*private* String artist;

*private* int releaseYear;

*public* *AudioItems*(String title, String artist, double initialPrice, int releaseYear) {

super(title, initialPrice);

this.*artist* = artist;

this.*releaseYear* = releaseYear;

}

*public* int *getReleaseYear*() {

*return* releaseYear;

}

*public* String *artist*() {

*return* artist;

}

*public* String *toString*() {

NumberFormat cost = NumberFormat.*getCurrencyInstance*();

*return* super.*toString*() + " (" + artist + ")\tCost: " + cost.*format*(super.*getInitialPrice*());

}

}

**Dvd.java:**

import *java*.*text*.*NumberFormat*;

*public* *class* Dvd *extends* Item{

*public* *Dvd*(String title, double price) {

super(title, price);

}

*public* double *getSellingPrice*() {

*return* super.*getInitialPrice*();

}

*public* String *toString*() {

NumberFormat cost = NumberFormat.*getCurrencyInstance*();

*return* super.*toString*() + "\tCost: " + cost.*format*(*getSellingPrice*());

}

}

**Record.java:**

*public* *class* Record *extends* AudioItems {

*public* *Record*(String title, String artist, double initialPrice, int releaseYear) {

super(title, artist, initialPrice, releaseYear);

}

*public* double *getSellingPrice*() {

*return* (super.*getInitialPrice*()) \* ((2024 - super.*getReleaseYear*()) / 4.0);

}

}

**Cassette.java:**

*public* *class* Cassette *extends* AudioItems {

*public* *Cassette*(String title, String artist, double initialPrice, int releaseYear) {

super(title, artist, initialPrice, releaseYear);

}

*public* double *getSellingPrice*() {

*return* (super.*getInitialPrice*()) + (super.*getInitialPrice*()) / ((2024 - super.*getReleaseYear*()) / 6.0);

}

}

**Catalogue.java:**  
  
import *java*.*util*.*ArrayList*;

*public* *class* Catalogue {

*private* double storeValue;

*private* ArrayList<Item> items;

*public* *Catalogue*(double storeValue) {

this.*storeValue* = storeValue;

items = *new* ArrayList<Item>();

}

*public* boolean *sellItem*(Item i) {

*if* (*searchItemBinary*(i) != -1) {

storeValue += i.*getSellingPrice*();

items.*remove*(i);

*return* true;

} *else* {

*return* false;

}

}

*public* boolean *buyItem*(Item i) {

*if* (storeValue >= i.*getInitialPrice*()) {

storeValue -= i.*getInitialPrice*();

items.*add*(i);

*return* true;

} *else* {

*return* false;

}

}

*public* int *searchItemLinear*(Item i) {

int index = 0;

*for* (Item item *:* items) {

*if* (item.*compareTo*(i) == 0) {

*return* index;

}

index++;

}

*return* -1;

}

*public* String *printCatalogue*() {

String catalogue = "";

*for* (Item item*:* items) {

catalogue += item + "\n";

}

*return* catalogue;

}

*public* void *selectSort*() {

*for*(int outer = 0; outer < items.*size*()-1; outer++) {

int min = outer;

*for*(int inner = outer+1; inner < items.*size*(); inner++) {

*if* (items.*get*(min).*compareTo*(items.*get*(inner)) > 0) {

min = inner;

}

}

Item temp = items.*get*(outer);

items.*set*(outer, items.*get*(min));

items.*set*(min, temp);

}

}

*public* int *searchItemBinary*(Item i) {

int start = 0;

int end = items.*size*()-1;

*selectSort*();

*while*(start <= end) {

int middle = (start+end)/2;

int difference = items.*get*(middle).*compareTo*(i);

*if* (difference == 0) {

*return* middle;

}

*if* (difference < 0) {

start = middle + 1;

}

*if* (difference > 0) {

end = middle - 1;

}

}

*return* -1;

}

*public* boolean *isSorted*() {

*for* (int i = 0; i < items.*size*() - 1; i++) {

*if* (items.*get*(i).*compareTo*(items.*get*(i + 1)) > 0) { // *If any item is greater than the next, the list is not sorted*

*return* false;

}

}

*return* true;

}

}

**Driver.java:**

*public class Driver {*

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

*Record record1 = new Record("Record1", "A", 120, 2022);*

*AudioItems record2 = new Record("Record2", "B", 150, 2024);*

*Cassette cassette1 = new Cassette("Record1", "C", 200, 2000);*

*Dvd dvd1 = new Dvd("Dvd1", 50);*

*Item dvd2 = new Dvd("Dvd1", 60);*

*Cassette cassette2 = new Cassette("Cassette2", "D", 100, 2000);*

*Catalogue catalogue1 = new Catalogue(1200);*

*Catalogue catalogue2 = new Catalogue(0);*

*// TEST CASE 1: Add 5 items to Catalogue*

*catalogue1.buyItem(dvd1);*

*catalogue1.buyItem(dvd2);*

*catalogue1.buyItem(record1);*

*catalogue1.buyItem(record2);*

*catalogue1.buyItem(cassette1);*

*System.out.println("Is the catalogue sorted? " + catalogue1.isSorted());*

*System.out.println();*

*// TEST CASE 2: Remove items from Catalogue until it is empty*

*catalogue1.sellItem(record1);*

*catalogue1.sellItem(record2);*

*catalogue1.sellItem(cassette1);*

*catalogue1.sellItem(dvd1);*

*catalogue1.sellItem(dvd2);*

*// TEST CASE 3: Remove an item which is not in the catalogue*

*catalogue1.sellItem(cassette2);*

*// TEST CASE 4: Add an item to catalogue when store does not have enough money to buy it*

*catalogue2.buyItem(cassette2);*

*catalogue1.buyItem(dvd1);*

*catalogue1.buyItem(dvd2);*

*catalogue1.buyItem(record1);*

*catalogue1.buyItem(record2);*

*catalogue1.buyItem(cassette1);*

*// TEST CASE 5: Print the catalogue*

*System.out.println(catalogue1.printCatalogue());*

*System.out.println(catalogue2.printCatalogue());*

*}*

*}*

**DemoDriverInterface.java:**import *java*.*util*.*Scanner*;

*public* *class* DemoInterfaceDriver {

*static* Scanner scan = *new* *Scanner*(System.*in*);

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

Catalogue catalogue = *new* *Catalogue*(500); // *starting with store value of a 500*

// *Adding a few items in the catalogue*

Record record1 = *new* *Record*("Record1", "A", 120, 2022);

AudioItems record2 = *new* *Record*("Record2", "B", 150, 2024);

catalogue.*buyItem*(record1);

catalogue.*buyItem*(record2);

int input = 0;

*do* {

*try* {

System.*out*.*println*("Select 1 to add, 2 to remove, or 3 to quit: ");

input = Integer.*parseInt*(scan.*nextLine*());

*if* (input == 1) { // *add*

System.*out*.*println*("Input title, cost, artist and year (separated by new line characters)");

boolean isAdded = catalogue.*buyItem*(*createRecordFromUserInput*());

*if*(isAdded) {

System.*out*.*println*("Successfully added\n");

} *else* {

System.*out*.*println*("Not enough balance\n");

}

} *else* *if* (input == 2) {

System.*out*.*println*("Input titlle, cost, artist and year (separated by new line characters)");

boolean isDeleted = catalogue.*sellItem*(*createRecordFromUserInput*());

*if*(isDeleted) {

System.*out*.*println*("Successfully deleted\n");

} *else* {

System.*out*.*println*("Record not found\n");

}

}

} *catch* (NumberFormatException nfe) {

System.*out*.*println*("\nAn error occured with input");

System.*out*.*println*("Input 1, 2, or 3 for commands");

System.*out*.*println*("Costs should be doubles and years should be integers\n");

}

} *while* (input != 3);

}

*public* *static* Record *createRecordFromUserInput*() {

String title = scan.*nextLine*();

Double cost = Double.*parseDouble*(scan.*nextLine*());

String artist = scan.*nextLine*();

int year = Integer.*parseInt*(scan.*nextLine*());

*return* *new* *Record*(title, artist, cost, year);

}

}