abstract class Product

{

private double price

private String[]tags

private String brand

private String name

public Product(double newPrice, String[]newTags, String newBrand, String newName)

{

price of product = newPrice

tags of products = newTags

brand of product = newBrand

name of product = newName

}

getPrice()

{

return price of product

}

setPrice(double newPrice)

{

price of product = newPrice

}

getTags()

{

return tags of products

}

setTags(String [] newTags)

{

tags of products = newTags

}

getBrand()

{

return brand of product

}

setBrand(String newBrand)

{

brand of product = newBrand

}

getName()

{

return name or product

}

setName(String newName)

{

name or product = newName

}

printTags()

{

for number of tags of product

{

print a tag

}

}

abstract changeProductInfo()

abstract printSpecifications()

toString()

{

print name

print price

print brand

}

}

----------------------------------------------------------------------------------------------------------------------------

class Laptop extends Product

{

private double screenSize

private int storage

private double batteryLife

private double processorSpeed

private int ram

public Laptop(double newPrice, String[]newTags, String newBrand, String newName,double screenSize, int storage, double batteryLife, double processorSpeed, int ram)

{

super(newPrice, newTags, newBrand, newName)

size of screen=screenSize

storage of laptop=storage

Life of battery=batteryLife

speed of processor=processorSpeed

ram of laptop=ram

}

getScreenSize()

{

return size of screen

}

setScreenSize(double newScreenSize)

{

size of screen= newScreenSize

}

getStorage()

{

return storage of laptop

}

setStorage(int newStorage)

{

storage of laptop = newStorage

}

getBatteryLife()

{

return batter life of laptop

}

setBatteryLife(double newBatteryLife)

{

battery life of laptop = newBatteryLife

}

getProcessorSpeed()

{

return speed of processor

}

setProcessorSpeed(double newProcessorSpeed)

{

battery life of laptop = newProcessorSpeed

}

getRam()

{

return RAM of laptop

}

setRam(int newRam)

{

RAM of laptop = newRam

}

changeProductInfo()

{

prompt for laptop parameters

use mutator methods to change product specifications

}

toString()

{

print super.toString

}

printSpecifications()

{

print screen size,battery life, amount ram, processor speed, storage size

}

}

----------------------------------------------------------------------------------------------------------------------------

class Computer extends Product

{

private int storage

private double processorSpeed

private int ram

public Computer (double newPrice, String[]newTags, String newBrand, String newName,int storage, double processorSpeed, int ram)

{

super(newPrice, newTags, newBrand, newName)

ram of computer= ram

Speed of processor= processorSpeed

RAM of computer=ram

}

getStorage()

{

return storage of computer

}

setStorage(int newStorage)

{

Storage of computer = newStorage

}

getProcessorSpeed()

{

return speed of processor

}

setProcessorSpeed(double newProcessorSpeed)

{

battery life of computer = newProcessorSpeed

}

getRam()

{

return RAM of computer

}

setRam(int newRam)

{

RAM of computer = newRam

}

changeProductInfo()

{

prompt for computer parameters

use mutator methods to change product specifications

}

toString()

{

print super.toString

}

printSpecifications()

{

print amount of ram, processor speed, storage size

}

}

----------------------------------------------------------------------------------------------------------------------------

class mouse extends Product

{

private String type

private int cordLengthRange

private int dpi

private int buttons

private double width

private double height

private double length

public Mouse(double newPrice, String[]newTags, String newBrand, String newName, String type, int cordLengthRange, int dpi, int buttons, double width, double height, double length )

{

super(newPrice, newTags, newBrand, newName)

type of mouse= type

cord length or range= cordLengthRange

dpi of mouse=dpi

amount of buttons=buttons

width of mouse= width

height of mouse= height

length of mouse=length

}

getType()

{

return type of mouse

}

setType(String newType)

{

type = newType()

}

getCordLengthorRange()

{

return cordlengthorRange

}

setCordLength(int newCordLengthorRange)

{

cordLengthorRange = newCordLengthorRange

}

getDpi()

{

return dpi

}

setDpi(int newDpi)

{

dpi = newDpi

}

getButtons()

{

return buttons

}

setButtons(int newButton)

{

buttons = newButton

}

getWidth()

{

return width

}

setWidth(double newWidth)

{

width = newWidth

}

getHeight()

{

return height

}

setHeight(double newHeight)

{

height = newHeight

}

getDepth()

{

return depth

}

setDepth(double newDepth)

{

height = newDepth

}

changeProductInfo()

{

prompt for mouse parameters

use mutator methods to change product specifications

}

toString()

{

return super classes ToString

}

printSpecifications()

{

print type, cordLengthRange, dpi, buttons, width, length, height, length

}

}

----------------------------------------------------------------------------------------------------------------------------

class Keyboard extends Product

{

private boolean wireless

private int cordLengthRange

private boolean design

public Keybaord(double newPrice, String[]newTags, String newBrand, String newName,boolean wireless,int cordLengthRange, boolean design)

{

super(newPrice, newTags, newBrand, newName)

if keyboard is wireless or not=wireless

length or cord or range of cord= cordLengthRange

if keyboard has a design or not= design

}

getWireless()

{

return true or false if wireless or not

}

setWireless(boolean newWireless)

{

wireless = newWireless

}

getCordLengthRange()

{

return length of cord or range of keyboard

}

setCordLength(int newCordLengthRange)

{

cordLengthRange = newCordLengthRange

}

getDesign()

{

return true or false if keyboard is ergonmic or not

}

setDesign(boolean newDesign)

{

design = newDesign

}

changeProductInfo()

{

prompt for keyboard parameters

use mutator methods to change product specifications

}

toString

{

super.toString

}

printSpecifications()

{

print yes for wireless,or no if wired, cord length/range, yes if ergonomic design, no if not

}

}

----------------------------------------------------------------------------------------------------------------------------

class Printer extends Product

{

private String type

private Boolean colourType

private double height

private double width

private double length

public Printer(double newPrice, String[]newTags, String newBrand, String newName,boolean colourType, double height, double width, double length )

{

super(newPrice, newTags, newBrand, newName)

if printer prints colour or not= colourType

height of printer= height

width of printer= width

length of printer=length

}

getType()

{

return type of printer

}

setType(String newType)

{

Type of printer = newType

}

getcolourType()

{

return true or false if colour or not

}

setcolourType(boolean newcolourType)

{

colour = newWireless

}

getWidth()

{

return width of printer

}

setWidth(double newWidth)

{

width of printer = newWidth

}

getLength()

{

return length of printer

}

setLength(double newLength)

{

length of printer = newLength

}

changeProductInfo()

{

prompt for printer parameters

use mutator methods to change product specifications

}

printSpecfications()

{

print type, color, width, length, depth

}

toString()

{

super().toString

}

----------------------------------------------------------------------------------------------------------------------------

class Inventory

{

private Product [] inventory

loadFile(String fileName)//ignore all io cases

{

Scanner input = new Scanner(new FileWriter(String fileName))

Inventory = blank array list

int number of products = input.nextInt()

String type = input.next()

for (number of products)

{

if (type == C)

{

get computer parameters from file

create new computer object

store it into inventory

}

else if (type == L)

{

get laptop parameters from file

create new laptop object

store it into inventory

}

else if (type == K)

{

get keyboard parameters from file

create new keyboard object

store it into inventory

}

else if (type == M)

{

get mouse parameters from file

create new mouse object

store it into inventory

}

else if (type == P)

{

get printer parameters from file

create new printer object

store it into inventory

}

}

}

saveFile(String fileName)//ignore io cases

{

Scanner input = new Scanner()

print number of items to file

for (every product)

{

if (product == computer)

{

print C

print using toString

print computer Specs

}

else if (product == laptop)

{

print L

print using toString

print laptop Specs

}

else if (product == keyboard)

{

print K

print using toString

print keyboard Specs

}

else if (product == mouse)

{

print M

print using toString

print mouse Specs

}

else if (product == printer)

{

print P

print using toString

print printer Specs

}

}

}

changeProductInfo() AS

{

if (this is a Laptop)

(laptop)this.changeProductInfo()

else if (this is a Keyboard)

(keyboard)this.changeProductInfo()

else if (this is a Printer)

(printer)this.changeProductInfo()

else if (this is a Mouse)

(printer)this.changeProductInfo()

else if (this is a Computer)

(printer)this.changeProductInfo()

}

searchTags(String tag) AS

{

sortTags()

use recursive binary search algorithm to return an array of items that match

}

searchPrice(double price) AS

{

sortLowToHigh()

use recursive binary search algorithm to return an array of items that match

}

searchName(String name) AS

{

sortName()

use recursive binary search algorithm to return the one item that matches

}

//helper method

//selection sort

sortHighToLow(boolean sortedLowToHigh)

{

if(sortedLowToHigh)

{

for (the first item to the middle)

{

Product temp = last product

last product = first product

first product = temp

}

}

else

{

use selection sort algorithm to sort from high to low

}

}

//selection sort

sortLowToHigh(boolean sortedHighToLow)

{

if(sortedHighToLow)

{

for (the first item to the middle)

// i think this would be significantly more efficient

{

Product temp = last product

last product = first product

first product = temp

}

}

else

{

use selection sort algorithm to sort from low to high

}

}

sortBrand()

{

use selection sort algorithm to sort by brand alphabetically

}

sortName()

{

use selection sort algorithm to sort by name alphabetically

}

addNewProduct()

{

prompt for what type of product

prompt for paramters

create product

add to list

}

removeNewProduct(String typeOfProduct, String nameOfProduct) AS

{

for (every product)

{

if (the product from the inventory’s type == type of product to be removed)

{

if(the product from inventory’s name = name or product)

{

remove it

}

}

}

// alternative possibility to just pass the name of the product and use a binary search, might take more work tho, since you have to sort it first

}

printAllProducts

{

for (every product in inventory)

{

print product

}

}

}

http://lichess.org/7twRFEoC