**Programing languages (BSc, 18) Java Lab 9**

**Task 1**

Create a divisors function which takes two positive integers as parameters and returns their divisors in a list data structure.

class Main

{

public static LinkedList<Integer> divisors(int num)

{

LinkedList<Integer> result = new LinkedList<Integer>();

for (int i = 1; i <= Math.sqrt(num); ++i)

{

if (num % i == 0)

{

result.add(i);

if (num / i != i)

{

result.add(num/i);

}

}

}

return result;

}

public static void main(String[] args)

{

System.out.println(divisors(10));

System.out.println(divisors(1));

System.out.println(divisors(2));

System.out.println(divisors(7));

System.out.println(divisors(124));

}

}

### Task 2

#### A)

Create a Book class representing a generic book. A Book has three fields: author (String), title (String), and pageCount (int). Create a PrintedBook and an EBook class extending from Book. A book's author and the title should only be accessible to Book, while pageCount should be accessible to its subclasses.

The Book class has two constructors:

1. A constructor that takes the author, title, and pageCount as parameters and sets the corresponding fields. Make sure that the length of author is at least two, while the title's length is four or greater. If either one is incorrect, throw an IllegalArgumentException
2. A constructor that takes no arguments. It calls the previously deffined ctor with the following values:
   * author: John Steinbeck
   * title: Of Mice and Men
   * pageCount: 107

Add a getShortName method to Book, that returns a string composed of the first two chars of the author, first four chars of the title, and the pageCount.

Write a main class that constructs two Books, one with each constructor, then prints their short names (getShortName) to the console.

<https://www.geeksforgeeks.org/overriding-in-java/?ref=lbp>

#### B)

A printed book can have two cover types: Softcover and Hardcover. Create an enum to store these values. The PrintedBook class stores it's cover type in a field.

A PrintedBook must be constructible with and without parameters.

* Without parameters: call the superclass's parameterless constructor, set the coverType to Hardcover, and add six to pageCount to account for the additional pages added in print.
* With parameters: take all four fields and set them. (author, title, pageCount, coverType)

The EBook class stores its fileSize in an int field. An EBook instance can only be constructed by providing these parameters: author, title, pageCount, fileSize.

Add a getPrice method to PrintedBook, and EBook, it calculates the books price in the following way:

* For Softcover printed books the price is equal to: pageCount \* 2
* For Hardcover printed books the price is equal to: pageCount \* 3
* For EBooks the price is equal to: pageCount + fileSize

In Main instantiate PrintedBook and EBook. Print their short names and prices.

#### C)

Supplement the Book class with a toString method. It returns the book's author, title and page count. For EBooks, this toString is sufficient. PrintedBook should override this method by extending its parents toString with its coverType.

Books often get referenced in articles for these use cases you need to create a reference catalog. Depending on the book type, it includes author, title, and the cited pages.

Add a createReference method to Book. createReference takes an article's title (String) and a start and end page for the citation, and returns a string reference pointing to a book.

The reference's format is as follows: "getShortName() [start page - end page] referenced by article: <article title> "

For printed books and e-books, the formatting is different. Override createReference in PrintedBook and EBook.

For printed book the format is: "super class's toString() [start page - end page] referenced by article: <article title> "

For digital books the reference format is: "super class's toString() (File size: <file size>) [start page - end page] referenced by article: <article title>"

When using digital source materials, it's a good idea to indicate the date of use. Overload the EBook class' createReference method. This new createReferece should take an article's title and the date of use. It returns a reference in this format: "super class's toString() (File size: <file size>) referenced by article: <article title>, file's use date: <date of use>"

#### D)

Write a isSameAuthor function, taking two Book references and returning whether the two books authors match. Call this function with a selection of Book and Book subclass instances.

class Book

{

private String author, title;

protected int pages;

public String getAuthor() { return author; }

public Book()

{

this.author = "John Steinbeck";

this.title = "Of Mice and Men";

this.pages = 107;

}

public Book(String author, String title, int pages)

{

if (author.length() < 2 || title.length() < 4)

{

throw new IllegalArgumentException();

}

this.author = author;

this.title = title;

this.pages = pages;

}

public String getShortName()

{

return author.substring(0, 1) + ": " + title.substring(0, 3) + "; " + pages;

}

//@Override

public String toString()

{

return author + ": " + title + ", pages: " + pages;

}

public String createReference(String article, int from, int to)

{

return getShortName() + " [" + from + "-" + to + "] referenced in article: " + article;

}

}

enum CoverType

{

Softcover,

Hardcover;

}

class PrintedBook extends Book

{

protected CoverType cover;

public PrintedBook()

{

this.pages += 6;

this.cover = CoverType.Hardcover;

}

public PrintedBook(String author, String title, int pages, CoverType cover)

{

super(author, title, pages + 6);

this.cover = cover;

}

public int getPrice()

{

if (cover == CoverType.Softcover)

{

return pages \* 2;

}

else

{

return pages \* 3;

}

}

//@Override

public String toString()

{

if (cover == CoverType.Softcover)

{

return super.toString() + " (softcover)";

}

else

{

return super.toString() + " (hardcover)";

}

}

@Override

public String createReference(String article, int from, int to)

{

return super.toString() + " [" + from + "-" + to + "] referenced in article: " + article;

}

}

class EBook extends Book

{

protected int PDFSize;

public EBook(String author, String title, int pages, int PDFSize)

{

super(author, title, pages);

this.PDFSize = PDFSize;

}

public int getPrice()

{

return pages + PDFSize;

}

@Override

public String createReference(String article, int from, int to)

{

return super.toString() + " (PDF size: " + PDFSize + ") [" + from + "-" + to + "] referenced in article: " + article;

}

//@Override // compile error

public String createReference(String article, String date)

{

return super.toString() + " (PDF size: " + PDFSize + ") referenced in article: " + article + ", accessing PDF date: " + date;

}

}

class Main

{

public static boolean isSameAuthor(Book book1, Book book2)

{

return book1.getAuthor().equals(book2.getAuthor());

}

public static void main(String[] args)

{

Book book1 = new Book();

System.out.println(book1.getShortName());

Book book2 = new Book("author", "Title", 100);

System.out.println(book2.getShortName());

System.out.println();

PrintedBook pbook1 = new PrintedBook();

System.out.println(pbook1.getShortName());

System.out.println(" price = " + pbook1.getPrice());

PrintedBook pbook2 = new PrintedBook("author", "Printed: Title", 100, CoverType.Softcover);

System.out.println(pbook2.getShortName());

System.out.println(" price = " + pbook2.getPrice());

System.out.println();

EBook ebook1 = new EBook("author2", "Digitalised: Title", 100, 12);

System.out.println(ebook1.getShortName());

System.out.println(" price = " + ebook1.getPrice());

System.out.println();

System.out.println(book1);

System.out.println(pbook1);

System.out.println(pbook2);

System.out.println(ebook1);

System.out.println();

//Book book3 = new PrintedBook();

//System.out.println("book3 price: " + book3.getPrice()); // compile error

System.out.println(book1.createReference("articlename", 10, 20));

System.out.println(pbook1.createReference("articlename", 10, 20));

System.out.println(ebook1.createReference("articlename", "2020/04/11"));

System.out.println(ebook1.createReference("articlename", 10, 20));

System.out.println();

System.out.println(isSameAuthor(book1, book2));

System.out.println(isSameAuthor(book2, pbook2)); // LSP

}

}