Exercise 1

* **Step 1**: Create a class that represents a circle shape. The class should be named as *Circle* and include the following three integer fields:
* *x* : the x coordinate
* *y* : the y coordinate
* *r* : the radius  
    
    
  Choose appropriate access modifiers for the fields so that they cannot be accessed by other classes.
* **Step 2**: Create methods that provide access and modification (getters and setters) to the *x*, *y* and *r* fields.
* Choose appropriate names that briefly describe the functionality of each method
* Choose appropriately the return type and access modifiers of your methods
* **Step 3**: Create a default *constructor* that initializes a Circle to x=y=0 and r=5
* **Step 3.1**: Create an overloaded *constructor* that initializes all three instance variables
* **Step 4**: Create a method *printCircleDetails* that prints the details of your circle in the following message  
  \* I'm a circle at point (x,y) with radius r  
  Where *x*,*y* and *r* are the values of the current circle
* **Step 5**: Create a new class named *TestCircles* in the same package (directory) as your *Circle* class.
* Create a *main* method in your *TestCircles* class. In the *main* method:
* create a circle object,
* assign values (of your preference) to the object using the setter methods from *Step 2* and
* print the details of your circle using the *printCircleDetails* method from *Step 3*
* **Compile and run your program!**
* **Step 6**: Create an overloaded constructor which will initialize the x & y instance variables. The r instance variable should be initialized by calling the constructor from **Step 3**
* In your *main* method, create objects using each one of the (now three) available constructors and print (using the *print* method) your objects
* **Step 7**: Extend the functionality of your class by adding the following two methods:
* the *calculateArea* which will calculates and returns the area (π\*radius\*radius, π=3.14) and
* the *calculatePerimeter* which will calculates and returns the perimeter (2\*π\*radius, π=3.14)
* Choose the appropriate return types for your methods
* Test the functionality of the two new methods with the Circles that you created earlier in your main method
* **Step 8**: Add to your class a *static* and *final* field name *pi* (for π). Initialize its value upon declaring the field. Then, modify your *calculateArea* and *calculatePerimeter* methods of *Step 7* so that they make use of the new *pi* field.
* **Step 9**: Create a *copy constructor* for your *Circle* class.
* Create two circles where the second one will be a copy of the first one
* Print their details
* Change the values of the first circle and print again their details
* What do you observe? Explain your findings according to the *memory allocation* in slide #17
* **Step 10**: Create a method name *cocentric* that checks if two circles share the same *centre* (co-centric circles). The method should return *true* if circles are co-centric, or *false* if circles have different centres.
* What is the return type of the method?
* What type of argument should your method have?
* **Step 11**: Declare a field named *numberOfCreatedCircles* that is common to all instances of class Circle and counts the number of the created circles. Think carefully of:
* the type of your *numberOfCreatedCircles* field,
* any special modifiers that might need and,
* the place/places that the value of the field should be incremented

Exercise 2

* Create a *class diagram* (on paper or any other tool) for the relation among the following entities (Each entity is a *Class*):
* There is a *Car* that has a *CarLicense*. The *CarLicense* cannot exist without a *Car*.
* There is a *Driver* that owns (only one) *DriversLicense*. The *DriversLicense* cannot exist without a *Driver*.
* The *Driver* owns one or more *Car*s. The *Car* exists even without a *Driver*.
* Use the correct *UML connectors* for the above relations among the entities
* Create the skeleton source code for the above classes (no need for fields, constructors or methods) and compile it

Exercise 3

You have the following relations between entities:

* There is a *Library* that has a collection (single dimension array) of *Book*s
* Each *Book* has an *Author*
* The *Library* is operated by a *Librarian*
* The user can make requests regarding authors' and book availability only by asking the Librarian
* For all of the following classes, additional to the requirements described in the slides, create the **getter and setter methods** (only where is needed) for interacting with their instance variables and any necessary **constructors**
* **Class Author**
* Fields:
* **name**, of type *String*
* Methods:
* **toString**, return type String, returns the name of the author
* **Class Book**
* Fields:
* **title**, type String,
* **author**, type Author,
* **isbn**, type String,
* **physicalCopies**, type int,
* **availableCopies**, type int, and
* **timesRented**, type int
* **Important** :
* **isbn** cannot change after the initialization
* **Class Book**
* Methods:
* **toString**, return type String. Returns the details of the book including the Author details. The Authors' details should be acquired by the proper **toString** method
* **rentPhysicalCopy**, type *void*. Checks if there is an available copy for renting. If yes, then it prints a message of success. **What fields should be modified upon a successful rental?**
* **isAvailable**, return type boolean. Checks if there is at least one available physical copy of the book, and
* **hasAuthor**, return type boolean. Checks if a given name is the name of this book's author
* **Class Library**
* Fields:
* **books**, type Book[] (array of **Books**)
* Methods:
* **printAvailableBooks**, type *void*. Prints books that have at least one available physical copy. Hint: Use the **isAvailable** and the **toString** methods from the **Book** class
* **printBookDetails** (Searches for a book based on a given title. If the book exists then its details should be printed, otherwise an error message should be displayed)
* **Class Library**
* Methods:
* **printBooksFromAuthor**, type void. Prints only the books that have an author that matches a given name
* **printTheMostPopularBook**, type void. Prints the book with the highest number of the **timesRented** field.
* **Class Librarian**
* Fields:
* **library**, type **Library**. The library that he manages.
* Methods:
* **findMeBooksFromAuthor**, type void. Receives an author name and delegates the request to the library's **printBooksFromAuthor** method
* **findMeAvailableBooks**, type void. Delegates the request to the library's **printAvailableBooks** method
* **findMeBook**, type void. Receives a book's title and delegates the request to the library's **printBookDetails** method
* **findMostPopularBook**, type void. Delegates the request to the library's **printTheMostPopularBook** method
* Create a **TestLibrary** class with a **main** method in which you will execute the following code block:
* */\*\* Create Random authors \*/*
* Author ruth = **new** Author("Ruth");
* Author diane = **new** Author("Diane");
* Author jacqueline = **new** Author("Jacqueline");
* Author rachel = **new** Author("Rachel");
* Author joan = **new** Author("Joan");
* Author theresa = **new** Author("Theresa");
* Author angela = **new** Author("Angela");
* Author helen = **new** Author("Helen");
* Author lisa = **new** Author("Lisa");
* */\*\* Create Random books from the above authors \*/*
* Book book1 = **new** Book("Book1",ruth,"368777540-2",10,2,20);
* Book book2 = **new** Book("Book2",diane,"963099898-2",10,1,22);
* Book book3 = **new** Book("Book3",jacqueline,"005382097-2",10,0,23);
* Book book4 = **new** Book("Book4",rachel,"538310208-2",10,3,24);
* Book book5 = **new** Book("Book5",joan,"562448132-2",10,4,26);
* Book book6 = **new** Book("Book6",theresa,"670364563-2",10,2,21);
* Book book7 = **new** Book("Book7",angela,"466916869-2",10,5,17);
* Book book8 = **new** Book("Book8",helen,"764674973-2",10,0,15);
* Book book9 = **new** Book("Book9",lisa,"052469721-2",10,6,17);
* Book book10 = **new** Book("Book10",ruth,"609291817-2",10,3,13);
* Book book11 = **new** Book("Book11",diane,"451378028-2",10,8,12);
* Book book12 = **new** Book("Book12",jacqueline,"142352773-2",10,6,20);
* Book book13 = **new** Book("Book13",rachel,"115135166-2",10,0,20);
* Book book14 = **new** Book("Book14",joan,"631942468-2",10,3,20);
* Book book15 = **new** Book("Book15",theresa,"323662444-2",10,0,23);
* Book book16 = **new** Book("Book16",angela,"121360492-2",10,0,12);
* Book book17 = **new** Book("Book17",helen,"391199302-2",10,0,20);
* Book book18 = **new** Book("Book18",ruth,"549307784-2",10,1,20);
* Book book19 = **new** Book("Book19",ruth,"368777230-2",10,23,20);
* Book book20 = **new** Book("Book20",ruth,"793027213-2",10,0,20);
* */\*\* Add the books to a Book array\*/*
* Book[] books = {book1,book2,book3,book4,book5,book6,book7,
* book8,book9,book10,book11,book12,book13,book14,book15,
* book16,book17,book18,book19,book20};*/\*\* Assign the book collection to the library \*/*
* Library library = **new** Library(books);
* */\*\* Librarian, theGuyWhoKnowsAlot, undertakes the operation of the library \*/*
* Librarian theGuyWhoKnowsAlot = **new** Librarian(library);
* theGuyWhoKnowsAlot.findMeAvailableBooks();
* theGuyWhoKnowsAlot.findMeBook("Book3");
* theGuyWhoKnowsAlot.findMeBooksFromAuthor("Ruth");

theGuyWhoKnowsAlot.findMostPopularBook();

* **Compare your output to the one in the next slide**
* The expected output (format can vary between different implementations) after the running the code from the previous slide is:

The following books are available at the library **for** renting

Books available **for** renting:

1. Book [title=Book1, author=Ruth, isbn=368777540-2, physicalCopies=10, availableCopies=2, timesRented=20]

2. Book [title=Book2, author=Diane, isbn=963099898-2, physicalCopies=10, availableCopies=1, timesRented=22]

3. Book [title=Book4, author=Rachel, isbn=538310208-2, physicalCopies=10, availableCopies=3, timesRented=24]

4. Book [title=Book5, author=Joan, isbn=562448132-2, physicalCopies=10, availableCopies=4, timesRented=26]

5. Book [title=Book6, author=Theresa, isbn=670364563-2, physicalCopies=10, availableCopies=2, timesRented=21]

6. Book [title=Book7, author=Angela, isbn=466916869-2, physicalCopies=10, availableCopies=5, timesRented=17]

7. Book [title=Book9, author=Lisa, isbn=052469721-2, physicalCopies=10, availableCopies=6, timesRented=17]

8. Book [title=Book10, author=Ruth, isbn=609291817-2, physicalCopies=10, availableCopies=3, timesRented=13]

9. Book [title=Book11, author=Diane, isbn=451378028-2, physicalCopies=10, availableCopies=8, timesRented=12]

10. Book [title=Book12, author=Jacqueline, isbn=142352773-2, physicalCopies=10, availableCopies=6, timesRented=20]

11. Book [title=Book14, author=Joan, isbn=631942468-2, physicalCopies=10, availableCopies=3, timesRented=20]

12. Book [title=Book18, author=Ruth, isbn=549307784-2, physicalCopies=10, availableCopies=1, timesRented=20]

13. Book [title=Book19, author=Ruth, isbn=368777230-2, physicalCopies=10, availableCopies=23, timesRented=20]

Book with title= 'Book3' found! Details:

Book [title=Book3, author=Jacqueline, isbn=005382097-2, physicalCopies=10, availableCopies=0, timesRented=23]

Book with author= 'Ruth' found! Details:

Book [title=Book1, author=Ruth, isbn=368777540-2, physicalCopies=10, availableCopies=2, timesRented=20]

Book with author= 'Ruth' found! Details:

Book [title=Book10, author=Ruth, isbn=609291817-2, physicalCopies=10, availableCopies=3, timesRented=13]

Book with author= 'Ruth' found! Details:

Book [title=Book18, author=Ruth, isbn=549307784-2, physicalCopies=10, availableCopies=1, timesRented=20]

Book with author= 'Ruth' found! Details:

Book [title=Book19, author=Ruth, isbn=368777230-2, physicalCopies=10, availableCopies=23, timesRented=20]

Book with author= 'Ruth' found! Details:

Book [title=Book20, author=Ruth, isbn=793027213-2, physicalCopies=10, availableCopies=0, timesRented=20]

Most popular book:

Book [title=Book5, author=Joan, isbn=562448132-2, physicalCopies=10, availableCopies=4, timesRented=26]

Book with title:'Book0' not found

Book with author:'angor' not found

Exercise 4 (BONUS!)

Create a command-line user interface for Exercise 3, providing the user, at least, the following options:  
Welcome to the Bootcamp library.  
How do you want to proceed?  
1. Show all available books  
2. Search for a book (by book title)  
3. Display books from a specific author  
4. Show me the most popular book  
q. Quit  
Create (a fully detailed) **class diagram** for the entities described in exercise 3.