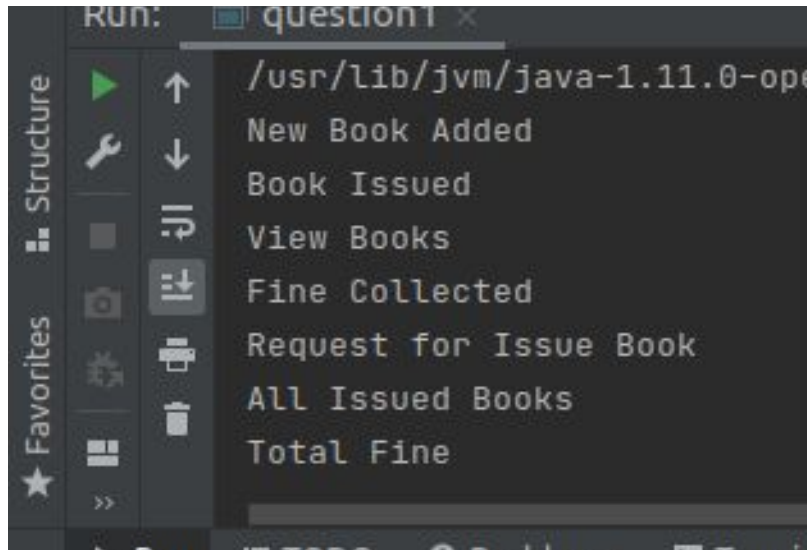


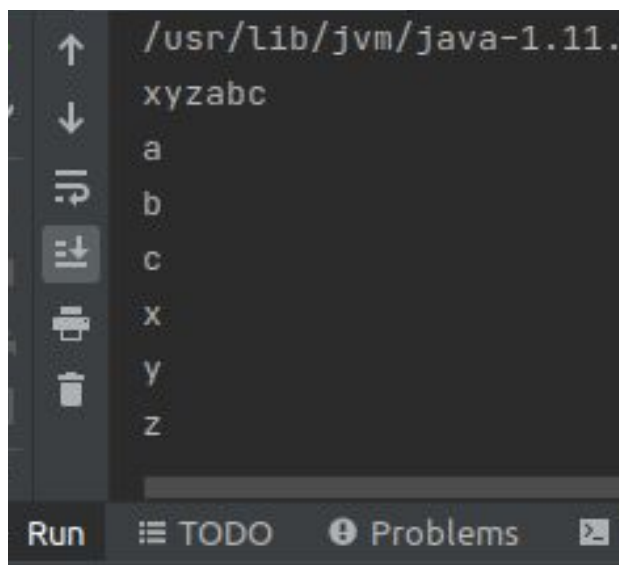
1. Create Java classes having suitable attributes for Library management system. Use OOPs concepts in your design. Also try to use interfaces and abstract classes.



A screenshot of a Java IDE's Run console window. The window title is 'Run: question1 x'. The console output shows a list of library management operations: '/usr/lib/jvm/java-1.11.0-op...', 'New Book Added', 'Book Issued', 'View Books', 'Fine Collected', 'Request for Issue Book', 'All Issued Books', and 'Total Fine'. The IDE interface includes a 'Structure' tab on the left and a 'Run' button at the bottom.

```
Run: question1 x
/usr/lib/jvm/java-1.11.0-op
New Book Added
Book Issued
View Books
Fine Collected
Request for Issue Book
All Issued Books
Total Fine
```

2. WAP to sorting string without using string Methods?.



A screenshot of a Java IDE's Run console window. The window title is 'Run: xyzabc'. The console output shows the result of sorting the string 'xyzabc' without using string methods: 'xyzabc', 'a', 'b', 'c', 'x', 'y', and 'z'. The IDE interface includes a 'Run' button at the bottom.

```
Run: xyzabc
xyzabc
a
b
c
x
y
z
```

3. WAP to produce NoClassDefFoundError and ClassNotFoundException exception.

```
question13.java 17 | data.showData();
question3 x
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intellij-idea-community/273/lib/idea_rt.jar=41473
Exception in thread "main" java.lang.NoClassDefFoundError: Create breakpoint : sample
    at java.base/java.lang.ClassLoader.defineClass1(Native Method)
    at java.base/java.lang.ClassLoader.defineClass(ClassLoader.java:1017)
    at java.base/java.security.SecureClassLoader.defineClass(SecureClassLoader.java:174) <5 internal calls>
    at java.base/jdk.internal.loader.BuiltinClassLoader.defineClass(BuiltinClassLoader.java:800)
    at java.base/jdk.internal.loader.BuiltinClassLoader.findClassOnClassPathOrNull(BuiltinClassLoader.java:698)
    at java.base/jdk.internal.loader.BuiltinClassLoader.loadClassOrNull(BuiltinClassLoader.java:621)
es are up-to-date (moments ago)
```

```
question16.java
question3_b x
/usr/lib/jvm/java-1.11.0-openjdk-amd64/
ClassNotFoundException: NoName

Process finished with exit code 0
```

4. WAP to create singleton class.

```
question11
question4 x
/usr/lib/jvm/java-1.11.0-openjdk-amd6
A object value PRIVATE CONSTRUCTOR
B object value PRIVATE CONSTRUCTOR

Process finished with exit code 0
```

5. WAP to show object cloning in java using cloneable and copy constructor both.

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intel-opencl/1.0.0-1/libexec/intel-opencl-1.0.0-1.jar -jar ...
Name: abhishek Age: 23
Name: abhishek Age: 23
Name: abhishek Age: 23

Process finished with exit code 0
```

6. WAP showing try, multi-catch and finally blocks.

```
16         }
17         catch(Exception e)
    question6 x
    /usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intel-opencl/1.0.0-1/libexec/intel-opencl-1.0.0-1.jar -jar ...
Arithmetic exception occurredjava.lang.ArithmeticException: / by zero
Exception Resolved

Process finished with exit code 0
```

7. WAP to convert seconds into days, hours, minutes and seconds.

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/snap/intel-opencl/1.0.0-1/libexec/intel-opencl-1.0.0-1.jar -jar ...
day: 258 hour: 8 minute: 53 seconds: 52

Process finished with exit code 0
```

8. WAP to read words from the keyboard until the word done is entered. For each word except done, report whether its first character is equal to its last character. For the required loop, use a

a)while statement

```
qwe
First and last character are not equal
qwq
First and last character are equal
done
```

b)do-while statement

```
wwe
First and last character are not equal
www
First and last character are equal
done

Process finished with exit code 0
```

9. Design classes having attributes for furniture where there are wooden chairs and tables, metal chairs and tables. There are stress and fire tests for each products.

```
pass
woodenchair
wooden
pass

Process finished with ex
```

10. Design classes having attributes and method(only skeleton) for a coffee shop. There are three different actors in our scenario and i have listed the different actions they do also below

* Customer

- Pays the cash to the cashier and places his order, get a token number back
- Waits for the intimation that order for his token is ready
- Upon intimation/notification he collects the coffee and enjoys his drink

(Assumption: Customer waits till the coffee is done, he wont timeout and cancel the order. Customer always likes the drink served. Exceptions like he not liking his coffee, he getting wrong coffee are not considered to keep the design simple.)

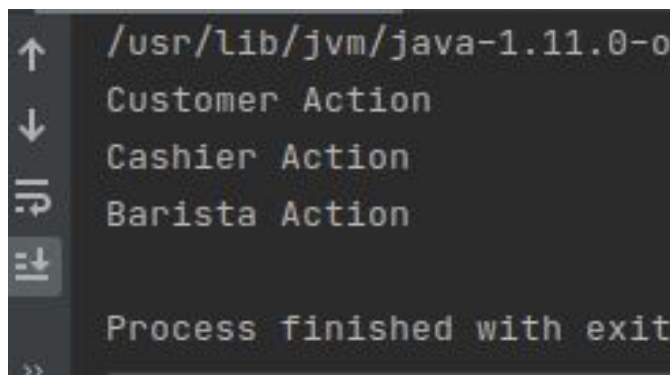
* Cashier

- Takes an order and payment from the customer
- Upon payment, creates an order and places it into the order queue
- Intimates the customer that he has to wait for his token and gives him his token

(Assumption: Token returned to the customer is the order id. Order queue is unlimited. With a simple modification, we can design for a limited queue size)

* Barista

- Gets the next order from the queue
- Prepares the coffee
- Places the coffee in the completed order queue
- Places a notification that order for token is ready



11. Convert the following code so that it uses nested while statements instead of for statements:

```
int s = 0;
int t = 1;
for (int i = 0; i < 10; i++)
{
    s = s + i;
    for (int j = i; j > 0; j--)
    {
        t = t * (j - i);
    }
}
```

```

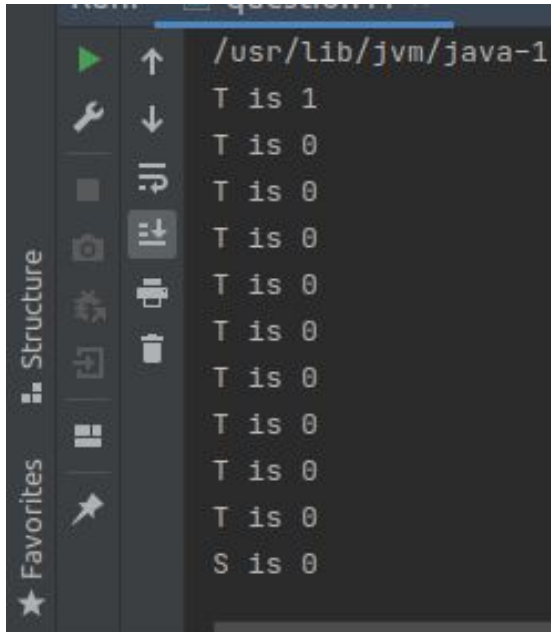
s = s * t;

System.out.println("T is " + t);

}

System.out.println("S is " + s);

```



12.What will be the output on new Child(); ?

```

class Parent extends Grandparent {

    {
        System.out.println("instance - parent");
    }

    public Parent() {
        System.out.println("constructor - parent");
    }

    static {
        System.out.println("static - parent");
    }

}

```

```
class Grandparent {

    static {
        System.out.println("static - grandparent");
    }
    {
        System.out.println("instance - grandparent");
    }
    public Grandparent() {
        System.out.println("constructor - grandparent");
    }
}

class Child extends Parent {
    public Child() {
        System.out.println("constructor - child");
    }
    static {
        System.out.println("static - child");
    }
    {
        System.out.println("instance - child");
    }
}
```

```
↑ /usr/lib/jvm/java-1.11.0-openj
↓ static - grandparent
static - parent
static - child
instance - grandparent
constructor - grandparent
instance - parent
constructor - parent
instance - child
constructor - child
```

Q13. Create a custom exception that do not have any stack trace.

```
↑ /usr/lib/jvm/java-1.11.0-openjdk-amd6
↓ myage: Invalid age

Process finished with exit code 0
```