2022503003

Date: 27/09/2024

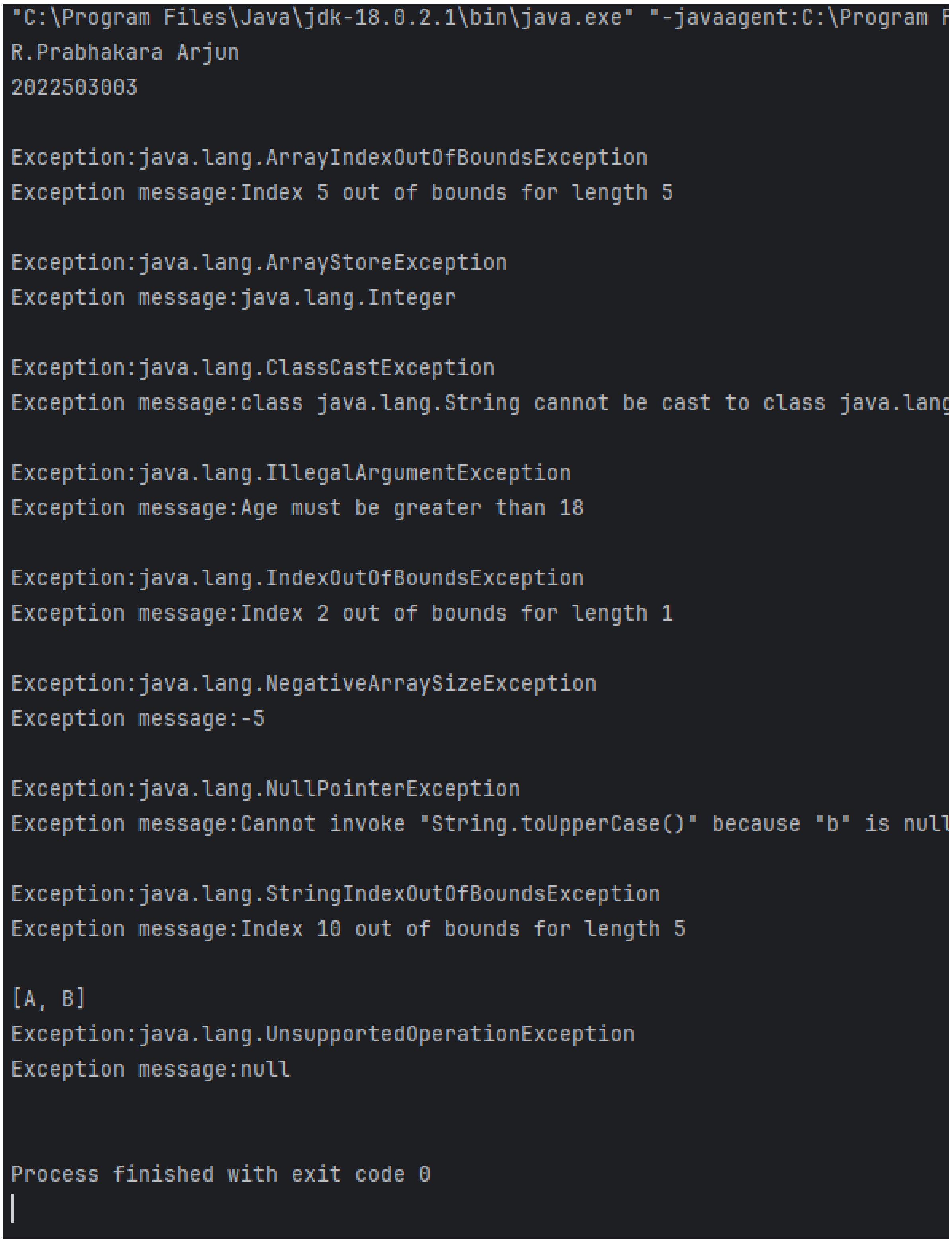
1. Write a program to perform unchecked exception. Use appropriate try-catch blocks to handle these exceptions and provide meaningful error messages

Code:

```
import java.sql.SQLOutput;
import java.util.ArrayList;
import java.util.List;

public class differentTypesOfException {
    static void  checkAge(int a){
        if(a<18){
            throw new IllegalArgumentException("Age must be greater than 18");
        }
    }
    public static void main(String[] args){
        System.out.println("R.Prabhakara Arjun\n2022503003\n");
        Integer[] arr=new Integer[5];
        try{
            int a=arr[5];
            System.out.println();
        } catch(IndexOutOfBoundsException e) {
            System.out.println("Exception:"+e.getClass().getName());
            System.out.println("Exception message:"+e.getMessage());
            System.out.println();
        }
        try{
            Object[] x=new String[3];
            x[0]=1;
        }
        catch(ArrayStoreException e){
            System.out.println("Exception:"+e.getClass().getName());
            System.out.println("Exception message:"+e.getMessage());
            System.out.println();
        }
        try{
           Object a="hello";
           Integer b=(Integer)a;
        }
        catch(ClassCastException e){
            System.out.println("Exception:"+e.getClass().getName());
            System.out.println("Exception message:"+e.getMessage());
            System.out.println();
        }
        try{
            checkAge(15);
```

```
}
catch(IllegalArgumentException e){
    System.out.println("Exception:"+e.getClass().getName());
    System.out.println("Exception message:"+e.getMessage());
    System.out.println();
}
try{
    ArrayList<Integer> arr1=new ArrayList<>();
    arr1.add(1);
    System.out.println(arr1.get(2));
}
catch(IndexOutOfBoundsException e){
    System.out.println("Exception:"+e.getClass().getName());
    System.out.println("Exception message:"+e.getMessage());
    System.out.println();
}
try{
    String[] arr2=new String[-5];
}
catch(NegativeArraySizeException e){
    System.out.println("Exception:"+e.getClass().getName());
    System.out.println("Exception message:"+e.getMessage());
    System.out.println();
}
try{
   String b=(null);
   b.toUpperCase();
}
catch(NullPointerException e){
    System.out.println("Exception:"+e.getClass().getName());
    System.out.println("Exception message:"+e.getMessage());
    System.out.println();
}
try{
    String her="abi!!";
    her.charAt(10);
}
catch(StringIndexOutOfBoundsException e){
    System.out.println("Exception:"+e.getClass().getName());
    System.out.println("Exception message:"+e.getMessage());
    System.out.println();
}
try{
    List<String> unmodifieable= List.of("A","B");
    System.out.println(unmodifieable);
    unmodifieable.add("c");
}
catch(UnsupportedOperationException e){
```

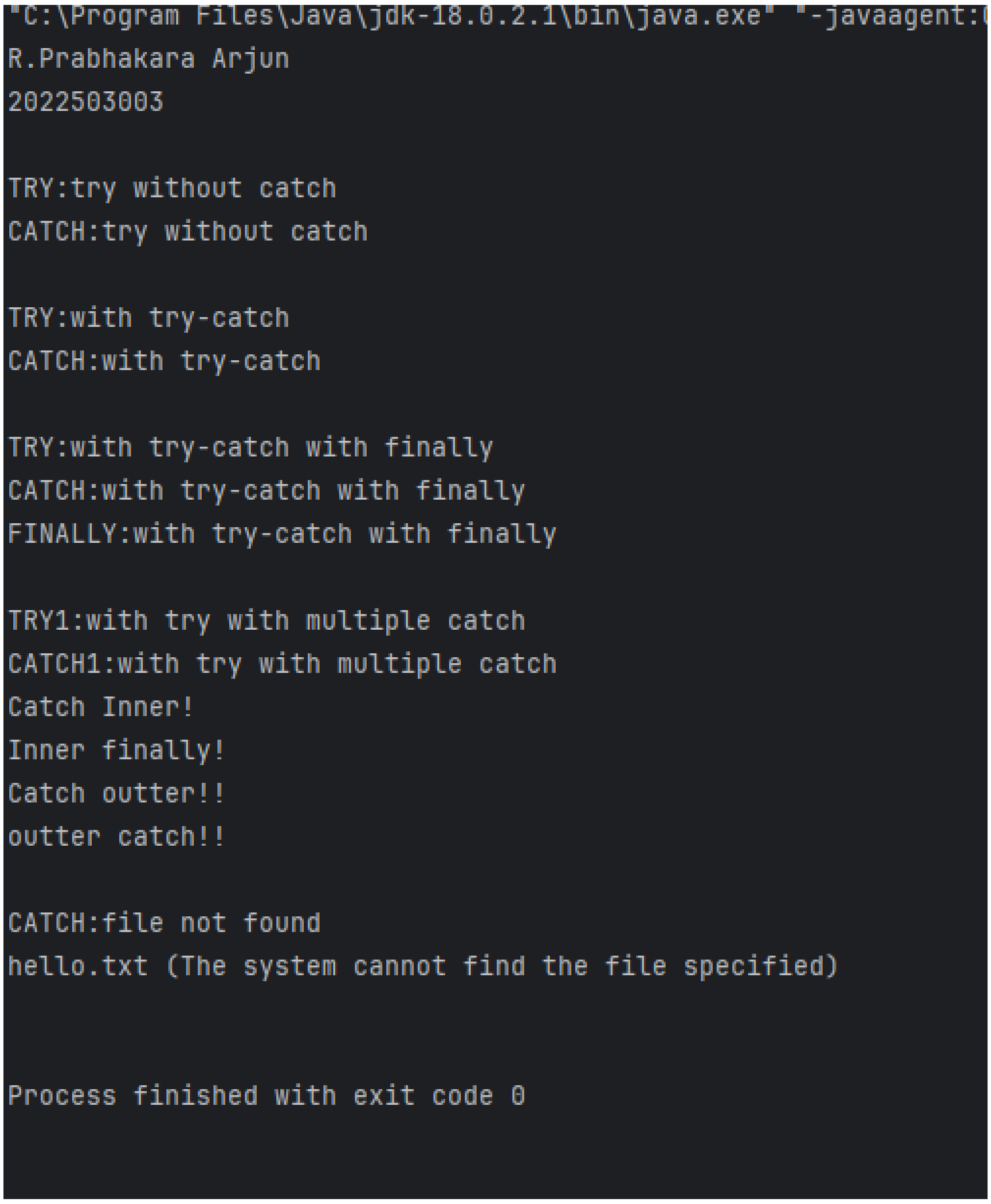
```
            System.out.println("Exception:"+e.getClass().getName());
            System.out.println("Exception message:"+e.getMessage());
            System.out.println();
        }
    }
}
```



```
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" "-javaagent:C:\Program F
R.Prabhakara Arjun
2022503003

Exception:java.lang.ArrayIndexOutOfBoundsException
Exception message:Index 5 out of bounds for length 5

Exception:java.lang.ArrayStoreException
Exception message:java.lang.Integer

Exception:java.lang.ClassCastException
Exception message:class java.lang.String cannot be cast to class java.lang

Exception:java.lang.IllegalArgumentException
Exception message:Age must be greater than 18

Exception:java.lang.IndexOutOfBoundsException
Exception message:Index 2 out of bounds for length 1

Exception:java.lang.NegativeArraySizeException
Exception message:-5

Exception:java.lang.NullPointerException
Exception message:Cannot invoke "String.toUpperCase()" because "b" is null

Exception:java.lang.StringIndexOutOfBoundsException
Exception message:Index 10 out of bounds for length 5

[A, B]
Exception:java.lang.UnsupportedOperationException
Exception message:null



Process finished with exit code 0
```

2. Write a program that demonstrates different try-catch-finally block combinations
a. Try without catch block Apps
b. Try without finally block
c. Try with catch and finally block
d. Try with multiple catch block
e. Nested try catch finally block
f. Try with resources

```
import javax.naming.AuthenticationNotSupportedException;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class combinationTryCatchFinal {
    public static void main(String[] args){
        System.out.println("R.Prabhakara Arjun\n2022503003\n");
        try {
            System.out.println("TRY:try without catch");
        }
        finally {
            System.out.println("CATCH:try without catch\n");
        }
        try{
            throw  new IOException("TRY:with try-catch");
        }
        catch (IOException E){
            System.out.println(E.getMessage());
            System.out.println("CATCH:with try-catch\n");
        }
        try{
            throw  new IOException("TRY:with try-catch with finally");
        }
        catch (IOException E){
            System.out.println(E.getMessage());
            System.out.println("CATCH:with try-catch with finally");
        }
        finally {
            System.out.println("FINALLY:with try-catch with finally\n");
        }

        try{
            throw new IOException("TRY1:with try with multiple catch");
            //throw new ArithmeticException("TRY1:with try with multiple catch");
            //unreachable state
        }
        catch (IOException E){
            System.out.println(E.getMessage());
    System.out.println("CATCH1:with try with multiple catch");
}
catch(ArithmeticException e){
    System.out.println(e.getMessage());
    System.out.println("CATCH2:with try with multiple catch\n");
}
try{
    try{
        int a=5/0;
    }
    catch(ArithmeticException e){
        System.out.println("Catch Inner!");
    }
    finally {
        System.out.println("Inner finally!");
    }
    throw new NullPointerException();
}
catch(Exception b){
    System.out.println("Catch outter!!");
}
finally {
    System.out.println("outter catch!!\n");
}

try(BufferedReader a=new BufferedReader(new FileReader("hello.txt"))){
    System.out.println("File found!");
}
catch(IOException O){
    System.out.println("CATCH:file not found");
    System.out.println(O.getMessage());
    System.out.println();
}}}
```

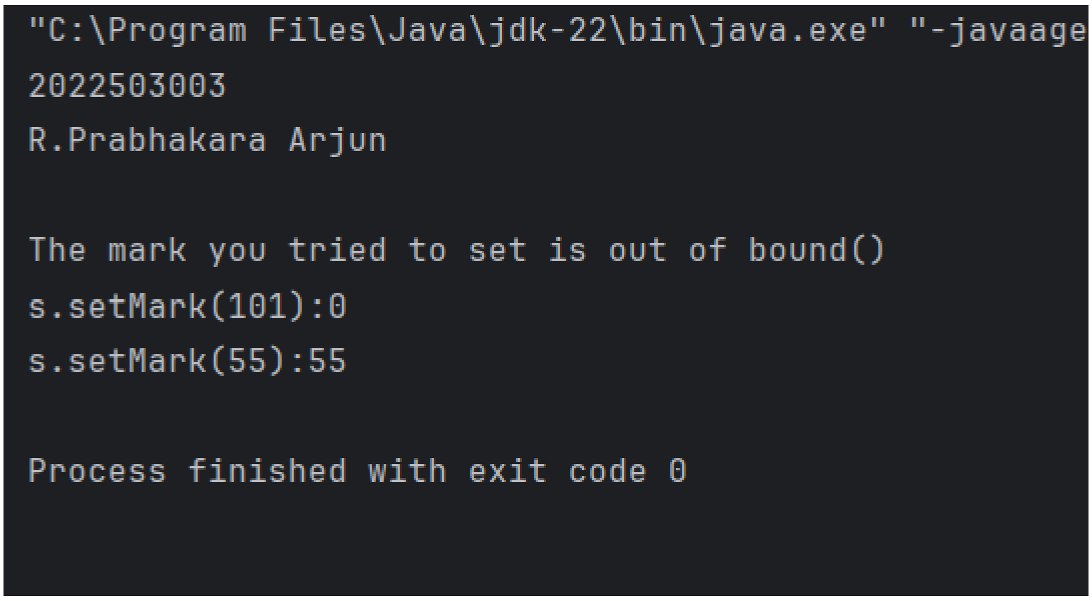


```
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" "-javaagent:
R.Prabhakara Arjun
2022503003

TRY:try without catch
CATCH:try without catch

TRY:with try-catch
CATCH:with try-catch

TRY:with try-catch with finally
CATCH:with try-catch with finally
FINALLY:with try-catch with finally

TRY1:with try with multiple catch
CATCH1:with try with multiple catch
Catch Inner!
Inner finally!
Catch outter!!
outter catch!!

CATCH:file not found
hello.txt (The system cannot find the file specified)


Process finished with exit code 0
```

3. Create a custom exception class called InvalidMarkException that extends Exception. Then, write a Student class with a method to set marks that throws this custom exception if the mark is out of range (e.g., less than 0 or greater than 100).

<u>Code:</u>

```
class invalidMarkException extends Exception{
    public invalidMarkException(String message){
        super(message);
    }
}
class Student{
    int marks;
    void setMark(int mark) throws invalidMarkException{
        if(mark<0 || mark>100){
            throw new invalidMarkException("The mark you tried to set is out of bound()");
        }
        marks=mark;
    }
    int getMark(){
        return marks;
    }
}
public class markHandling {
    public static void main(String[] args){
        System.out.println("2022503003\nR.Prabhakara Arjun\n");
        Student s=new Student();
        try {
            s.setMark(101);
        }catch(invalidMarkException e){
            System.out.println(e.getMessage());
        }
        System.out.println("s.setMark(101):"+s.getMark());
        s.getMark();
        try {
            s.setMark(55);
        }catch(invalidMarkException e){
            System.out.println(e.getMessage());
        }
        System.out.println("s.setMark(55):"+s.getMark());
    }
}
```

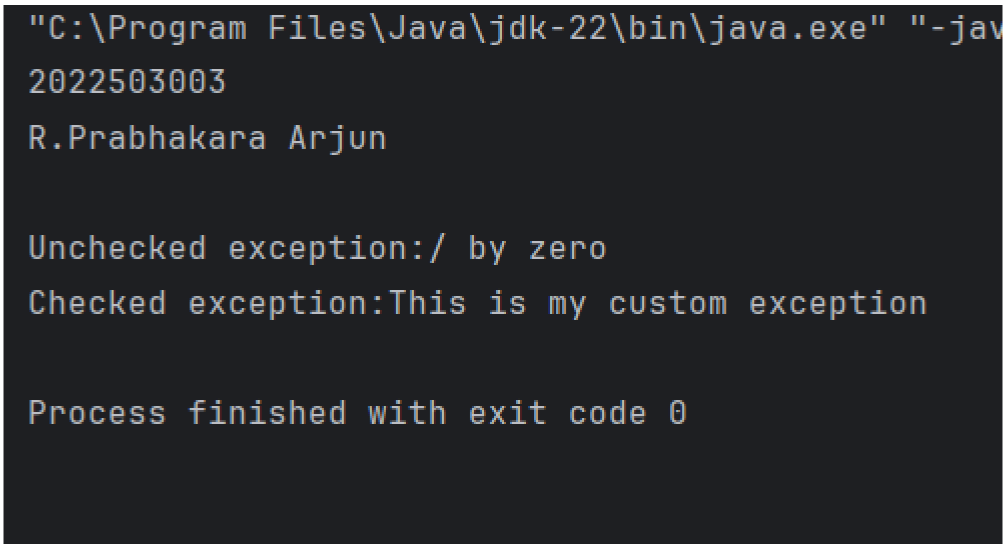


```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaage
2022503003
R.Prabhakara Arjun

The mark you tried to set is out of bound()
s.setMark(101):0
s.setMark(55):55

Process finished with exit code 0
```

4. Write a program to illustrate the propagation of checked and unchecked exception.

Code:

```
class myException extends Exception{
    myException(String msg){
        super(msg);
    }
}
public class propagationChechedUnchecked {
    static void checkedException() throws myException{
        throw new myException("This is my custom exception");
    }
    static void uncheckedException(){
        int result=10/0;
        System.out.println("Result"+result);
    }
    public static void main(String[] args) {
        System.out.println("2022503003\nR.Prabhakara Arjun\n");
        try{
            uncheckedException();
        }catch(Exception e){
            System.out.println("Unchecked exception:"+e.getMessage());
        }
        try{
            checkedException();
        }
        catch(Exception e){
            System.out.println("Checked exception:"+e.getMessage());
        }
    }
}
```

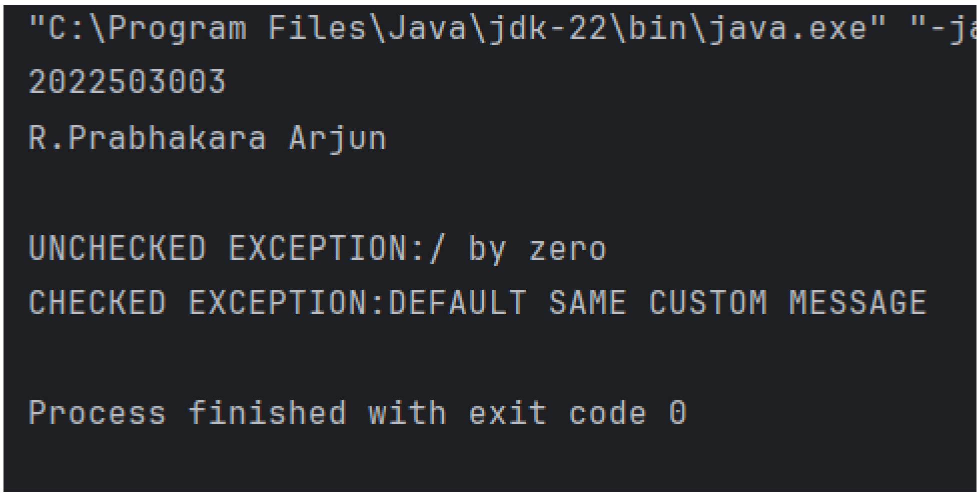


```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-jav
2022503003
R.Prabhakara Arjun

Unchecked exception:/ by zero
Checked exception:This is my custom exception

Process finished with exit code 0
```

5. Write a program to illustrate the method overloading in exception handling mechanism for checked and unchecked exception

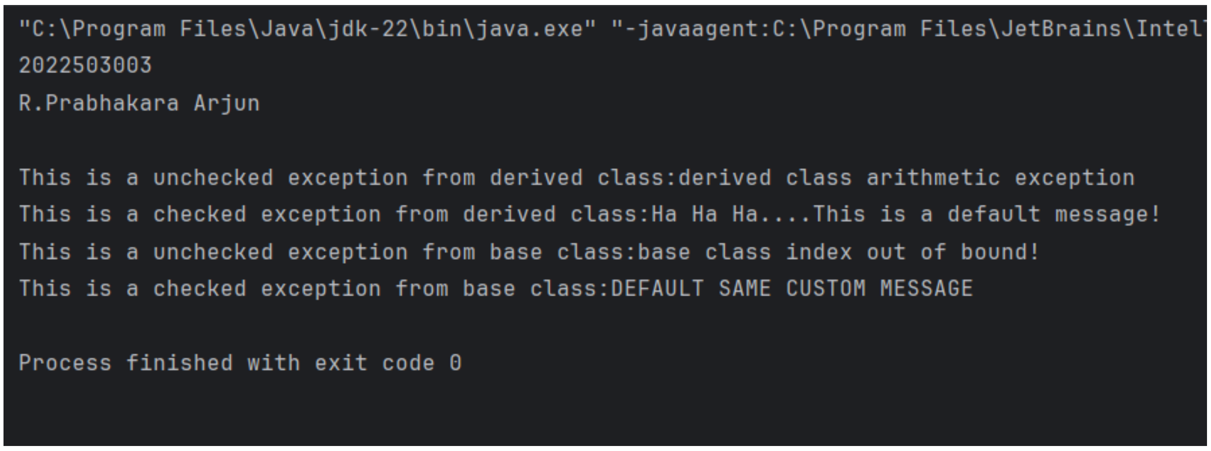
Code:

```
class myCustomException extends Exception{
    myCustomException(){
        super("DEFAULT SAME CUSTOM MESSAGE");
    }
}
class checker{
    void method(int num){
        int result=num/0;
        System.out.println("Result"+result);
    }
    void method(String msg) throws myCustomException{
        throw new myCustomException();
    }
}
public class methodOverloadingCheckedUnchecked {
    public static void main(String[] args) {
        checker check=new checker();
        System.out.println("2022503003\nR.Prabhakara Arjun\n");
        try{
            check.method(5);
        }catch(Exception e){
            System.out.println("UNCHECKED EXCEPTION:"+e.getMessage());
        }
        try{
            check.method("Abi!!");
        } catch (myCustomException e) {
            System.out.println("CHECKED EXCEPTION:"+e.getMessage());
        }
    }
}
```


```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-ja
2022503003
R.Prabhakara Arjun

UNCHECKED EXCEPTION:/ by zero
CHECKED EXCEPTION:DEFAULT SAME CUSTOM MESSAGE

Process finished with exit code 0
```

6. Implement a base class and a derived class to demonstrate exception handling in method overriding:
a) Overriding a method that throws an unchecked exception
b) Overriding a method that throws a checked exception

Code:
```
class myCustomException2 extends Exception{
    myCustomException2(){
        super("Ha Ha Ha....This is a default message!");
    }
}
class baseClass{
 void unChecked(){
     throw new IndexOutOfBoundsException("base class index out of bound!");
 }
 void checked() throws myCustomException,myCustomException2{
     throw new myCustomException();
 }
}
class dervidedClass extends baseClass{
    void unChecked(){
        throw new ArithmeticException("derived class arithmetic exception");
    }
    void checked() throws myCustomException2{
        throw new myCustomException2();
    }
}
public class overrideCheckedUnchecked {
    public static void main(String[] args) {
        System.out.println("2022503003\nR.Prabhakara Arjun\n");
        baseClass derive = new dervidedClass();
        try {
            derive.unChecked();
        }catch (Exception e){
            System.out.println("This is a unchecked exception from derived class:"+e.getMessage());
        }
        try {
            derive.checked();
        }
        catch(Exception e){
            System.out.println("This is a checked exception from derived class:"+e.getMessage());
        }
        baseClass base=new baseClass();
        try {
            base.unChecked();
        }catch (Exception e){
            System.out.println("This is a unchecked exception from base class:"+e.getMessage());
        }
        try {
            base.checked();
        }
        catch(Exception e){
            System.out.println("This is a checked exception from base class:"+e.getMessage());
        }
    }
}
```



```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\Intel
2022503003
R.Prabhakara Arjun

This is a unchecked exception from derived class:derived class arithmetic exception
This is a checked exception from derived class:Ha Ha Ha....This is a default message!
This is a unchecked exception from base class:base class index out of bound!
This is a checked exception from base class:DEFAULT SAME CUSTOM MESSAGE

Process finished with exit code 0
```