**Java Assignment**

Q1  
import java.util.Stack;

public class Evaluate

{

public static int eval(String expression)

{

char[] value = expression.toCharArray();

Stack<Integer> values = new Stack<Integer>();

Stack<Character> ops = new Stack<Character>();

for (int i = 0; i < value.length; i++)

{

if (value[i] == ' ')

continue;

if (value[i] >= '0' && value[i] <= '9')

{

StringBuffer sbuf = new StringBuffer();

while (i < value.length && value[i] >= '0' && value[i] <= '9')

sbuf.append(value[i++]);

values.push(Integer.parseInt(sbuf.toString()));

}

else if (value[i] == '(')

ops.push(value[i]);

else if (value[i] == ')')

{

while (ops.peek() != '(')

values.push(operator(ops.pop(), values.pop(), values.pop()));

ops.pop();

}

else if (value[i] == '+' || value[i] == '-' ||

value[i] == '\*' || value[i] == '/' || value[i] == '%')

{

while (!ops.empty() && precedence(value[i], ops.peek()))

values.push(operator(ops.pop(), values.pop(), values.pop()));

ops.push(value[i]);

}

}

while (!ops.empty())

values.push(operator(ops.pop(), values.pop(), values.pop()));

return values.pop();

}

public static boolean precedence(char op1, char op2)

{

if (op2 == '(' || op2 == ')')

return false;

if ((op1 == '\*' || op1 == '/') && (op2 == '+' || op2 == '-'))

return false;

else

return true;

}

public static int operator(char op, int b, int a)

{

switch (op)

{

case '+':

return a + b;

case '-':

return a - b;

case '\*':

return a \* b;

case '/':

if (b == 0)

throw new

UnsupportedOperationException("Cannot divide by zero");

return a / b;

case '%':

return a % b;

}

return 0;

}

public static void main(String[] args)

{

System.out.println(Evaluate.eval("8 \* 6 - 5"));

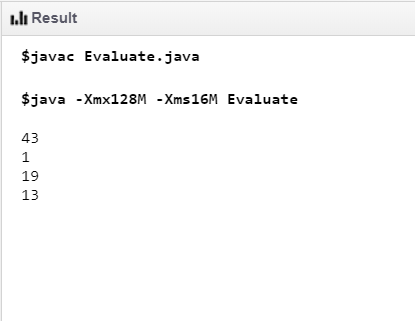
System.out.println(Evaluate.eval("( 55 + 9 ) % 9"));

System.out.println(Evaluate.eval("20 - ( 3 \* 5 ) / 8"));

System.out.println(Evaluate.eval("5 + 15 / 3 \* 2 - ( 8 % 3 )"));

}

}



Q2 American flag  
  
public class Main {

public static void main(String[] args) {

for (int i = 0; i < 4; i++) {

System.out.println("\* \* \* \* \* ===============================");

System.out.println(" \* \* \* \* \* ===============================");

}

System.out.println("\* \* \* \* \* ===============================");

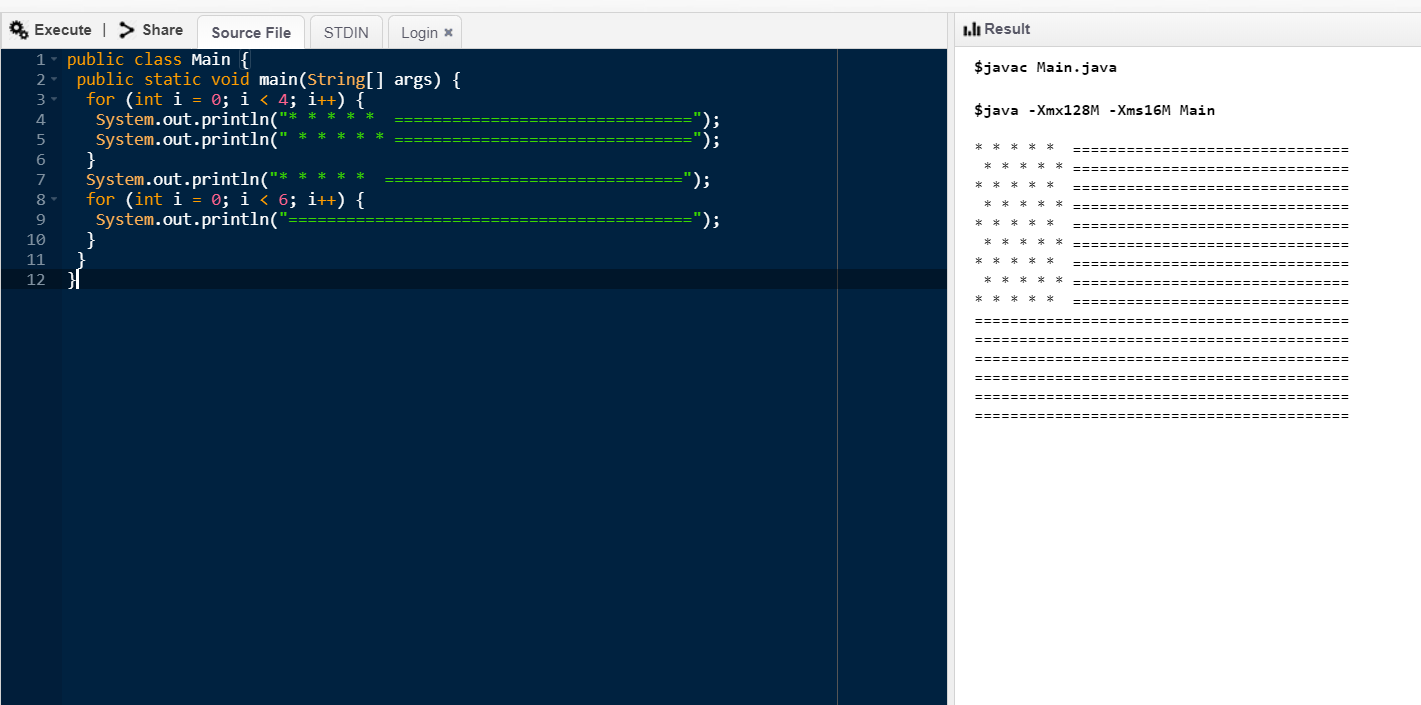
for (int i = 0; i < 6; i++) {

System.out.println("==========================================");

}

}

}



**HANDS ON ASSIGNMENT**

Q1.

public class Replace {

public static void main(String[] args)

{

String str = "Lion jumps over a dog to kill it.";

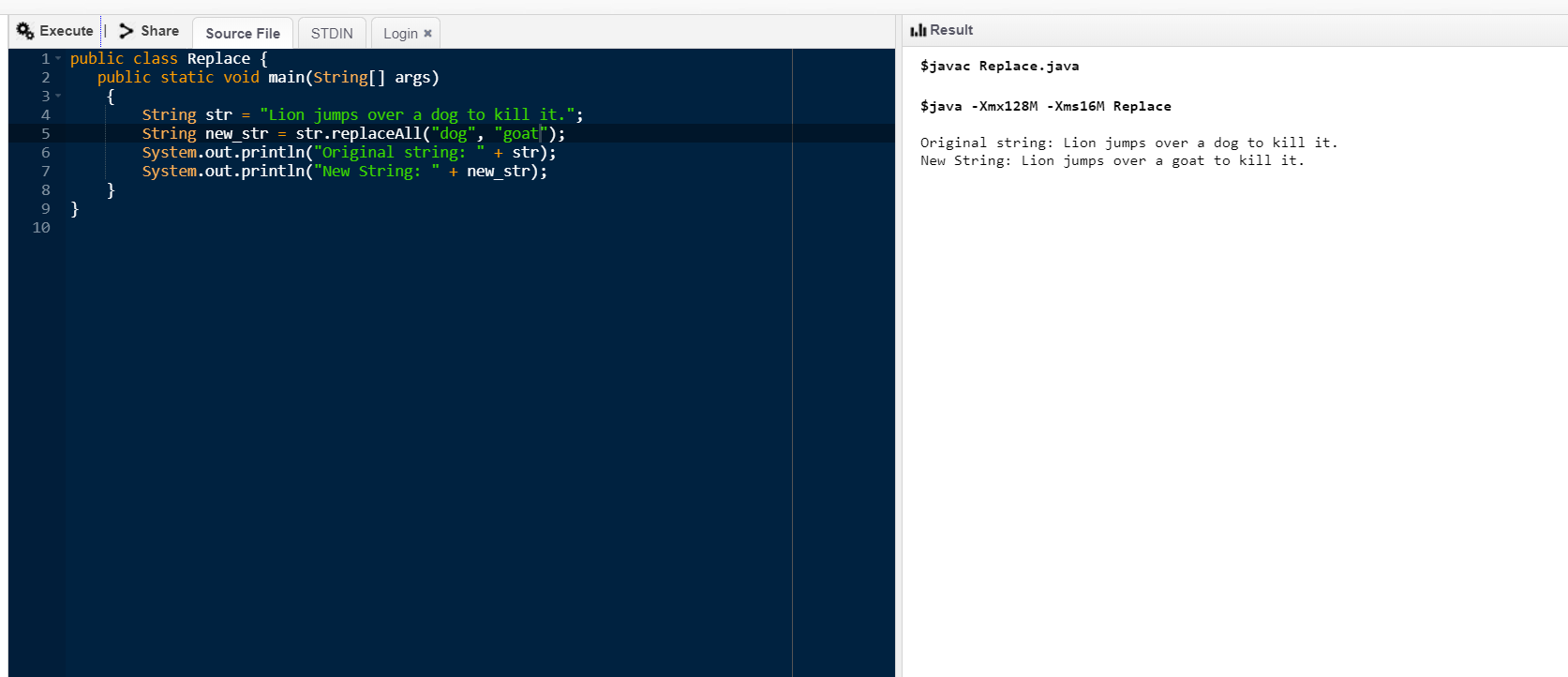
String new\_str = str.replaceAll("dog", "goat");

System.out.println("Original string: " + str);

System.out.println("New String: " + new\_str);

}

}

****

Q2.

import java.util.\*;

public class Reverse

{

public static void main(String[] args)

{

Map<String, String> colors = new HashMap<>();

colors.put("BLUE", "#0000FF");

colors.put("YELLOW", "#FFFF00");

colors.put("GREEN", "#008000");

colors.put("RED", "#FF0000");

Map<String, String> sortedMap = new LinkedHashMap<>();

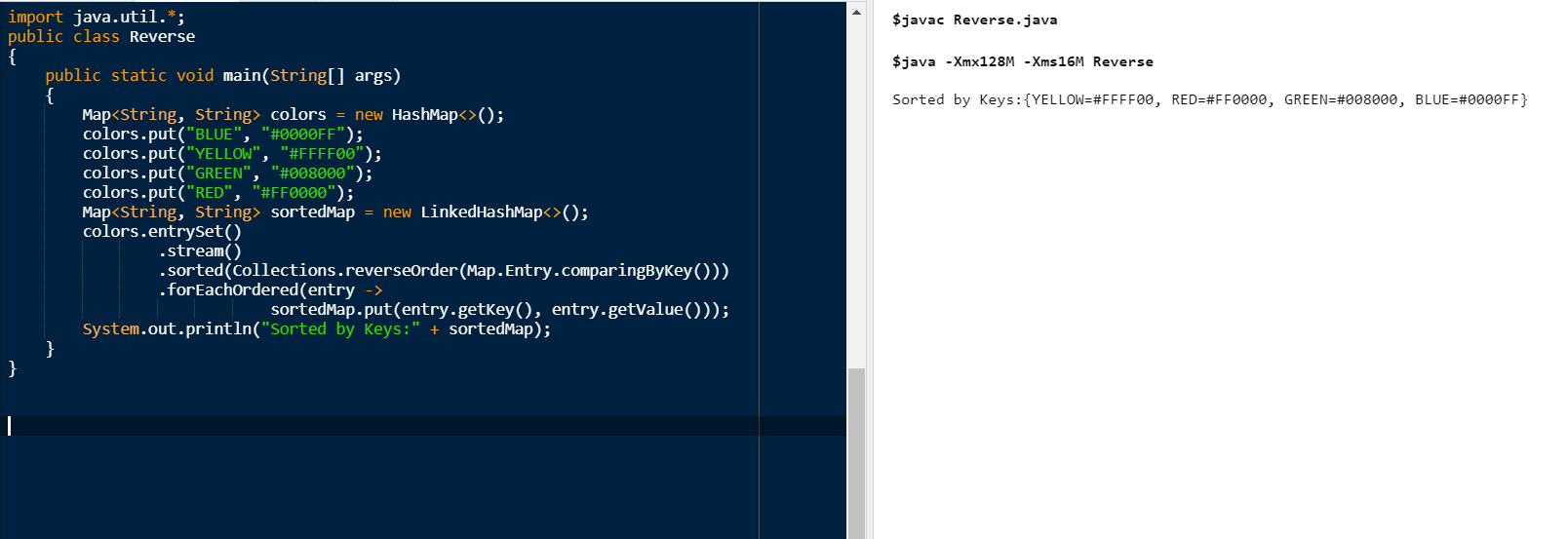
colors.entrySet().stream() .sorted(Collections.reverseOrder(Map.Entry.comparingByKey()))

.forEachOrdered(entry->sortedMap.put(entry.getKey(), entry.getValue()));

System.out.println("Sorted by Keys:" + sortedMap);

}

}



Q3. class PrimeException extends RuntimeException {

public PrimeException(String message) {

super(message);

}

}

class CheckPrime {

public void check() {

int i =0;

int num =0;

for (i = 1; i <= 100; i++)

{

try{

int counter=0;

for(num =i; num>=1; num--)

{

if(i%num==0)

{

counter = counter + 1;

}

}

if (counter ==2)

{

throw new PrimeException("Prime Number");

}

else {

System.out.println(i);

}

}

catch(PrimeException exp){

System.out.println(exp) ;

}

}

}

}

public class ExceptionPrime {

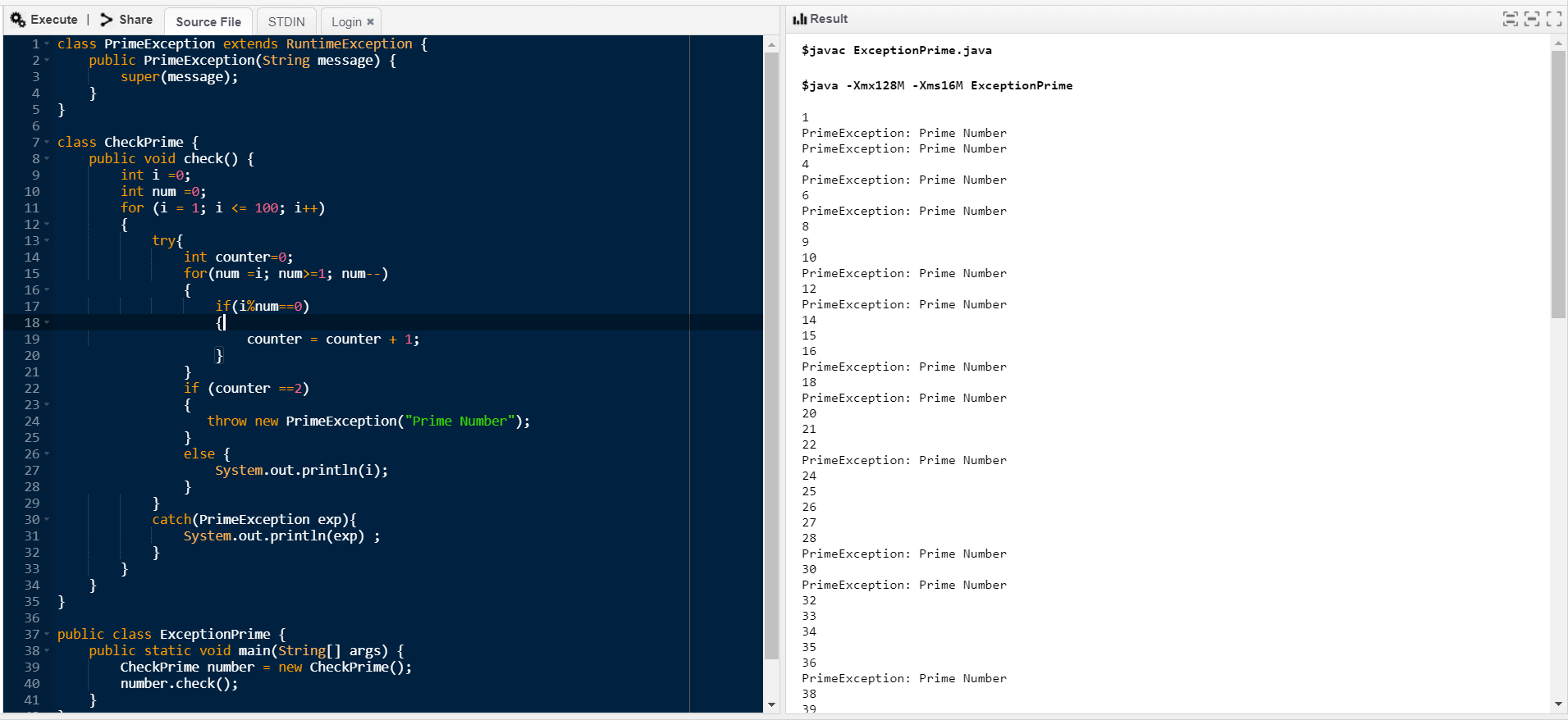
public static void main(String[] args) {

CheckPrime number = new CheckPrime();

number.check();

}

}



Q4. import java.io.\*;

class Emp implements Serializable {

private static final long serialversionUID = 129348938L;

transient int a;

static int b;

String name;

int age;

transient int c;

public Emp(String name, int age, int a, int b, int c)

{

this.name = name;

this.age = age;

this.a = a;

this.b = b;

this.c = c;

}

}

public class SerialExample {

public static void printdata(Emp object1)

{

System.out.println("name = " + object1.name);

System.out.println("age = " + object1.age);

System.out.println("a = " + object1.a);

System.out.println("b = " + object1.b);

System.out.println("c = " + object1.c);

}

public static void main(String[] args)

{

Emp object = new Emp("ab", 20, 2, 1000, 5);

String filename = "pranvi.txt";

// Serialization

try {

FileOutputStream file = new FileOutputStream (filename);

ObjectOutputStream out = new ObjectOutputStream (file);

out.writeObject(object);

out.close();

file.close();

System.out.println("After Serialization");

printdata(object);

}

catch (IOException ex) {

System.out.println("IOException is caught");

}

object = null;

// Deserialization

try {

FileInputStream file = new FileInputStream (filename);

ObjectInputStream in = new ObjectInputStream (file);

object = (Emp)in.readObject();

in.close();

file.close();

System.out.println("After Deserialization");

printdata(object);

}

catch (IOException ex) {

System.out.println("IOException is caught");

}

catch (ClassNotFoundException ex) {

System.out.println("ClassNotFoundException" + "is caught");

}

}

}

