**Advance Java Assignment**

Q1. (i) Using Lambda  
interface Interface

{

void abstractFun(int x);

default void normalFun()

{

System.out.println("Hello");

}

}

public class Main

{

public static void main(String args[])

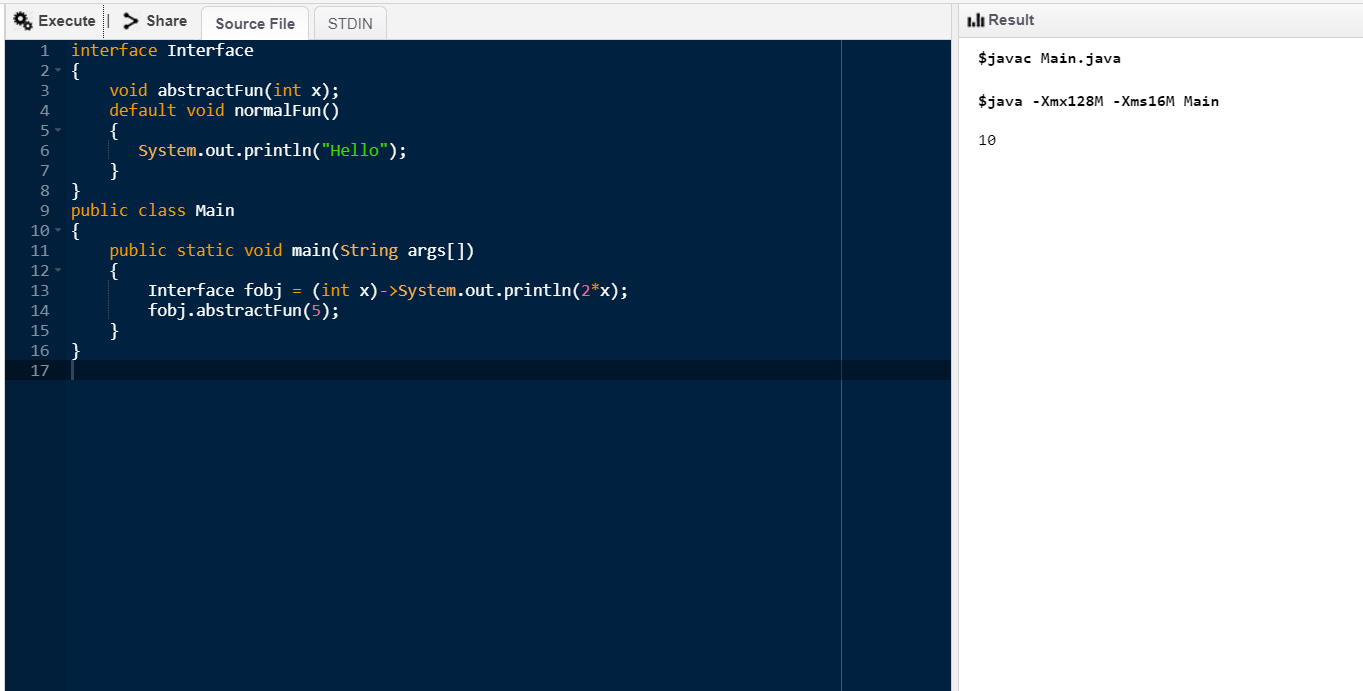
{

Interface fobj = (int x)->System.out.println(2\*x);

fobj.abstractFun(5);

}

}



(ii) Using static method  
interface Interface {

static void first()

{

System.out.println("Static Method");

}

void overrideMethod(String str);

}

public class InterfaceDemo implements Interface {

public static void main(String[] args)

{

InterfaceDemo interfaceDemo = new InterfaceDemo();

Interface.first();

interfaceDemo.overrideMethod("Override Method");

}

@Override

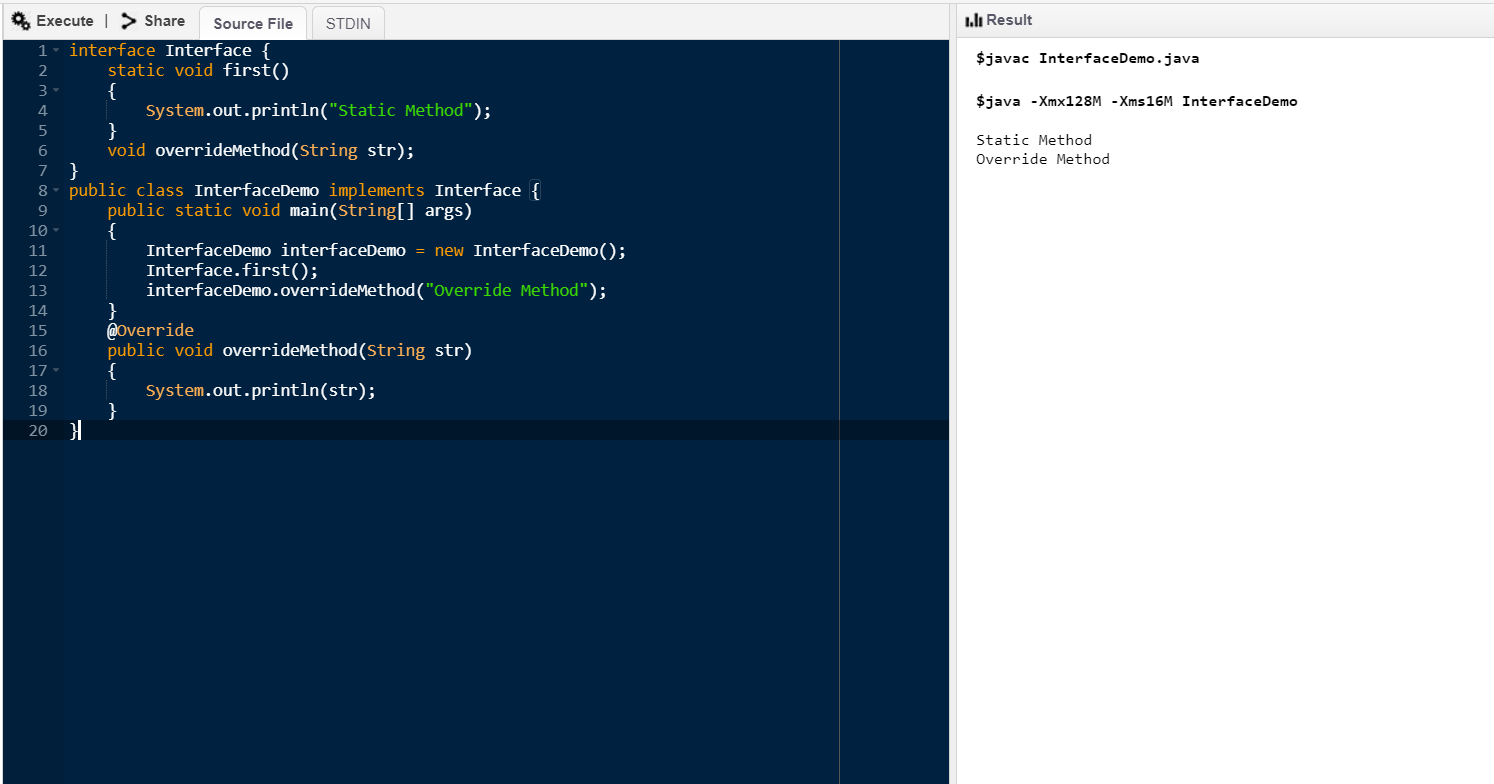
public void overrideMethod(String str)

{

System.out.println(str);

}

}



Q2.

interface FuncInterface

{

void abstractFun(int x);

}

public class Main

{

public static void main(String args[])

{

FuncInterface fobj = (x)->System.out.println(2\*x);

fobj.abstractFun(5);

}

}

Q3.

class Product {

private String name;

private Integer Category;

private Double price;

public Product(String name, Integer category, Double price) {

super();

this.name = name;

Category = category;

this.price = price;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Integer getCategory() {

return Category;

}

public void setCategory(Integer category) {

Category = category;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

}

Q4. We can use return statement for conditional rendering instead of continue as shown below.

public class without\_continue

{

public static void main(String[] args) {

List<Integer> num = Arrays.asList(1,2,3,4,5,6);

num.forEach( x-> {

if( x%2 == 0) {

return;

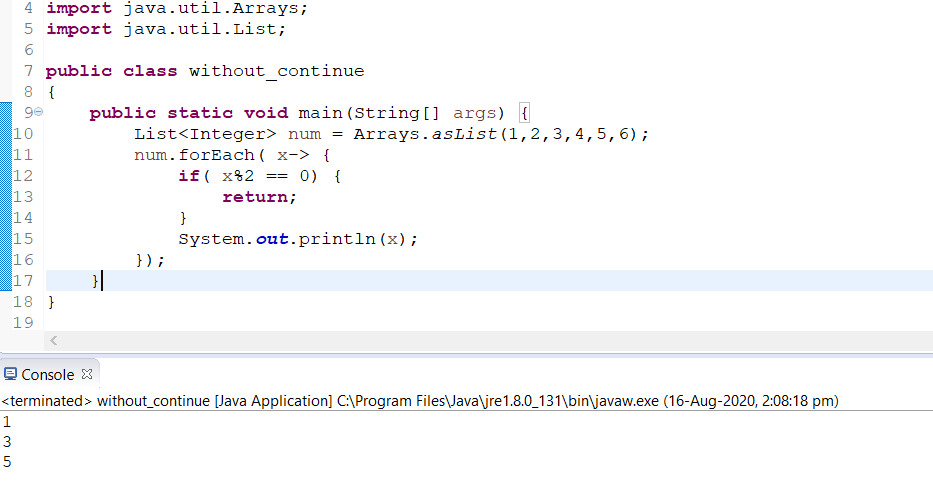
}

System.out.println(x);

});

}

}



Q5.

public class MySqlConnection

{

public static Connection doConnect()

{

Connection con=null;

try {

Class.forName("com.mysql.jdbc.Driver");

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

con=DriverManager.getConnection("jdbc:mysql://localhost/java\_project","root","bce");

System.out.println("Connected");

}

catch (ClassNotFoundException | SQLException e)

{

e.printStackTrace();

}

return con;

}

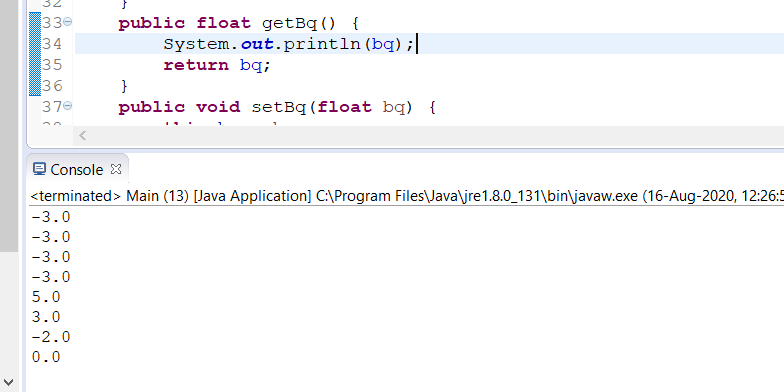
public static void main(String args[])

{

doConnect();

}

}



Q7. public class list\_files

{

static void RecursivePrint(File[] arr, int level)

{

for (File f : arr)

{

for (int i = 0; i < level; i++)

System.out.print("\t");

if(f.isFile())

System.out.println(f.getName());

else if(f.isDirectory())

{

System.out.println("[" + f.getName() + "]");

RecursivePrint(f.listFiles(), level + 1);

}

}

}

public static void main(String[] args)

{

String path = "C:\\Users\\Vivek\\Desktop\\Pranvi";

File directory = new File(path);

if(directory.exists() && directory.isDirectory())

{

File arr[] = directory.listFiles();

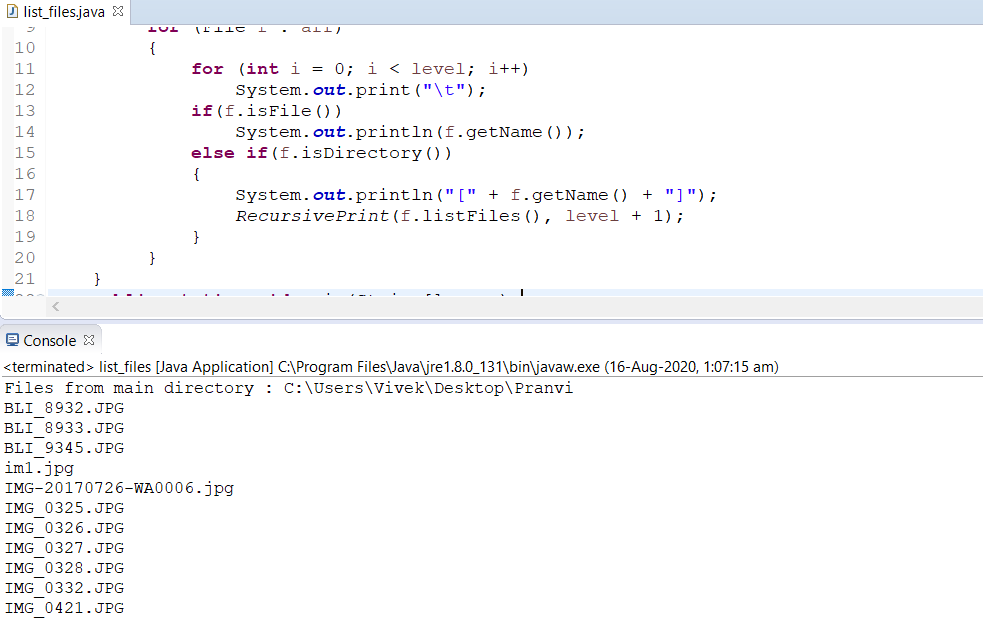
System.out.println("Files from main directory : " + directory);

RecursivePrint(arr, 0);

}

}

}



Q8.

public class txt\_files {

public static void main(String[] args) {

File directoryPath = new File("F:\\Linux files\\Documents\\docs\\Final\\Accolite\\Sessions\\Java");

File[] files=directoryPath.listFiles(new FilenameFilter() {

public boolean accept(File dir, String name) {

return name.endsWith(".txt");

}

});

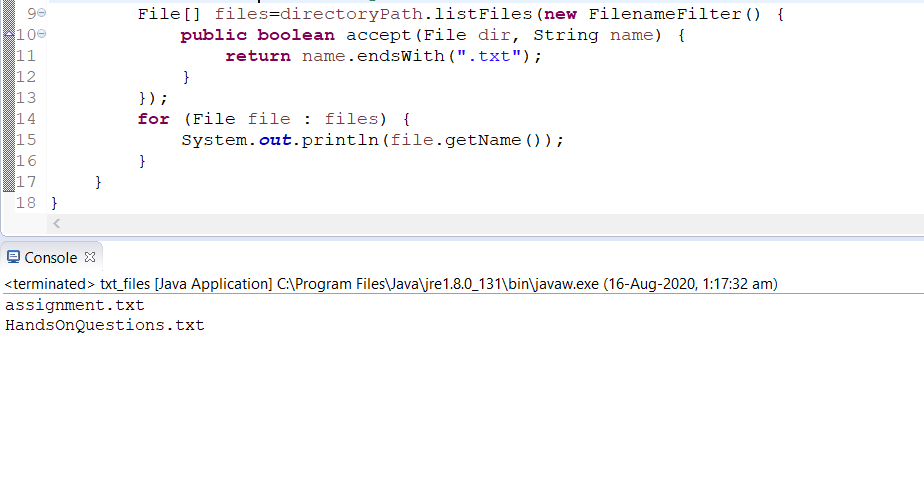
for (File file : files) {

System.out.println(file.getName());

}

}

}



Q9. public class file\_copy {

public static void main(String[] args) {

try {

FileReader fread = new FileReader("E:\\Real Java\\programs\\Exceptions\\Accolite\\src\\adv\_java\_assignment\\input.txt");

BufferedReader bread = new BufferedReader(fread);

FileWriter fwrite = new FileWriter("E:\\Real Java\\programs\\Exceptions\\Accolite\\src\\adv\_java\_assignment\\output.txt", true);

String s;

while ((s = bread.readLine()) != null) {

fwrite.write(s);

fwrite.flush();

}

bread.close();

fwrite.close();

System.out.println("file copied");

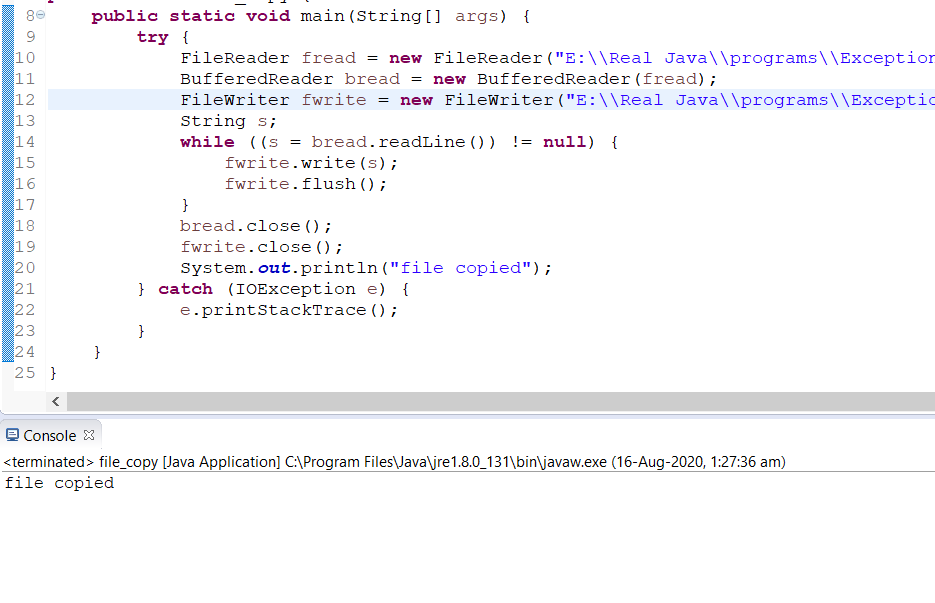
} catch (IOException e) {

e.printStackTrace();

}

}

}



Q10.

public class file\_transfer

{

public static void main(String[] args) throws IOException

{

Path temp = Files.move

(Paths.get("E:\\Real Java\\programs\\Exceptions\\Accolite\\src\\adv\_java\_assignment\\input.txt"),

Paths.get("E:\\Real Java\\programs\\Exceptions\\String\_class\\src\\inp.txt"));

if(temp != null)

{

System.out.println("File moved successfully");

}

else

{

System.out.println("Failed");

}

}

}

