Final 113–2021– online–S2

```
1.  Import java.util.*;
2.  class Excep_Test1{
3.   public static void main(String args[]){
4.  Scanner input=new Scanner (System.in);
5.  int x;
6.    try{
7.      try{
8.  // input from user
9.      x=input.nextInt();
10.      System.out.println("end of try block");
11.
12.     }catch(InputMismatchException e){x=10;}
13.
14.     try{
15.     int y[]=new int[x];
16.     y[8]=4;
17.     }catch(ArrayIndexOutOfBoundsException e){
18.        System.out.println("error in array index");}
19.
20.     System.out.println("The value of x is equal to " + x);
21.   }catch(Exception e){System.out.println("handled");}
22.
23. finally{ System.out.println("last statement");  }
24.
25.   System.out.println("normal flow..");
26.  }
27. }
```

Consider the following code :

What is the output of the following code, in case that the user enters '**a**'?

- end of try block
  The value of x is equal to 'a'
  normal flow..
- end of try block
  The value of x is equal to 10
  last statement
  normal flow..
- The value of x is equal to 10
  last statement
  normal flow..
- The value of x is equal to 10
  normal flow..

What is the output of the following code, in case that the user enter **5**?

- error in array index
  The value of x is equal to 5
  handled
  normal flow..
- end of try block
  error in array index
  last statement
  normal flow.
- end of try block
  error in array index
  The value of x is equal to 5
  last statement
  normal flow..
- error in array index
  The value of x is equal to 5
  last statement
  normal flow..

ESTION 3

Can you exchange the catch blocks in **line 17** and **line 20**?

- False, it will affect the order of the output
- False, it will cause a runtime error
- True
- False, it will cause a compilation error

Consider the following code:

```
1. class Excep_Test2{
2. public static void main(String args[]){
3. try{
4. int arr[]=new int[4];
5. arr[0]=10/0;
6. System.out.println("Last Statement of try block");
7.  }
8. catch(…….. e){
9. System.out.println("First Exception");
10. }
11.catch(…… e){
12.System.out.println("Second Exception ");
13. }
14.catch(…… e){
15.System.out.println("Third Exception ");
16.}
17.System.out.println("Out of the try-catch block");
18. }
19.}
```

the correct order for the catch blocks is

- ArithmeticException
  RuntimeException
  Exception

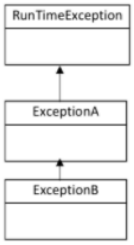
- ArithmeticException
  Exception
  RuntimeException

- RuntimeException
  Exception
  ArithmeticException

- Exception
  RuntimeException
  ArithmeticException

what is the output of the following code?

```
1 public class testExceptions {
2    static int f = 4;
3    public static int check1(int num) {
4       try {
5          if (num >  10)
6             throw new ExceptionA("Greater than 10");
7          }
8       catch (ExceptionB e) {
9           System.out.println("catch: " + e.getMessage());
10      }
11      finally {
12          f++;
13         System.out.println("finally: in check1");
14      }
15      return (f * num);
16   }
17   public static boolean check2(int num) {
18      try {
19         int multiple = check1(num);
20         if (multiple % 2 == 0)
21            throw new ExceptionB("The number is multiple of 2");
22         return true;
23      }
24      catch (ExceptionB e) {
25         System.out.println("check2 1st catch: " + e.getMessage());
26      }
27      catch (ExceptionA e) {
28         System.out.println("check2 2nd catch: " + e.getMessage());
29
30      }
31      finally {
32         System.out.println("check2 finally: f = " + f);
33         throw new IllegalArgumentException("check2 wrong number");
34      }    }
35      public static void main(String args[]) {
36      try {
37        if(check2(23))
38            throw new Exception("Wrong number");
39      }
40      catch (Exception e) {
41         System.out.println("main catch: " + e.getMessage());
42      } }
43 }
```



check2 2nd catch: Greater than 10
check2 finally: f = 5

- finally: in check1
  main catch: check2 wrong number

---

finally: in check1

- main catch: check2 wrong number

---

check2 2nd catch: Greater than 10
check2 finally: f = 5

- finally: in check1
  main catch: check2 wrong number

---

finally: in check1
check2 2nd catch: Greater than 10

- check2 finally: f = 5
  main catch: check2 wrong number

Consider the following code, starting from ***method1***, which is the correct call sequence in the stack trace?

```
public void method1()
{
try{
int a=5;
if(a>10) throw new Exception();
method3();
} catch (Exception e){….} }
```

```
public void method2() throws
Exception
{ int a=0;
try{
if (a==0)
throw new Exception();
method4();
}catch (Exception e) {…} }
```

```
public void method3() throws
Exception
{
try{
  method2();
throw new Exception();
}catch (Exception e){ … } }
```

```
public void method4()
{
System.out.println("The end of
trace");
}
```

- method 3
  method 2
  method 1
- method 4
  method 3
  method 2
  method 1
- method 3
  method 2
  method 1
  method 4
- method 2
  method 3
  method 1

STION 7

Are **method2** and **method3 propagator**?

- True
- False

STION 8

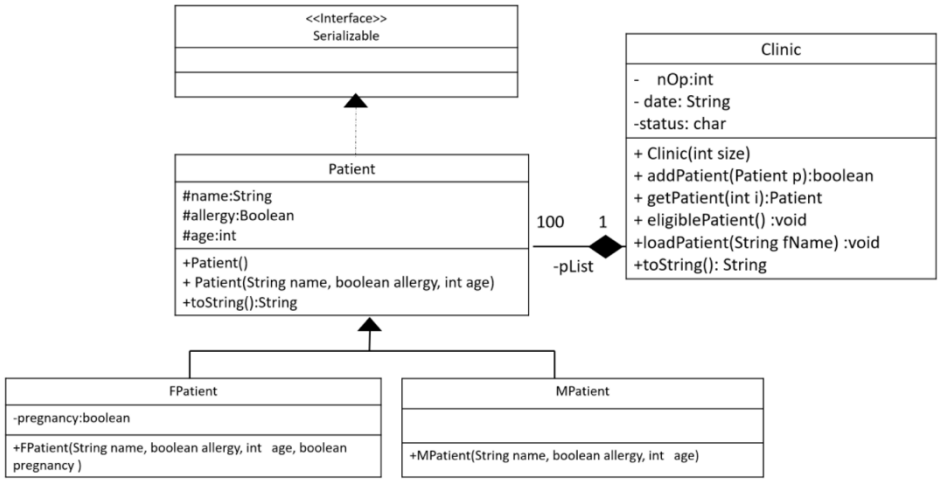
The print statement in **method4()** will never be called?

- True
- False

STION 9

Is **method1()** a catcher?

- True
- False



Consider the following implementation of **method loadPatient**, This method read objects of type **patient** one by one from an object file named **fName** Then adds them to **pList.**

```
1. public void loadPatient(String fName)            {
2. FileInputStream fileInput = new FileInputStream(new File(fName));
3. …… input = new ……
4. try {
5. while (true) {
6. Patient p = ……
7. if (p instanceof FPatient)
8. pList[nOp++] = new FPatient(p.name, p.allergy, p.age,((FPatient) p).isPregnent());
9. else
10.     pList[nOp++] = new MPatient(p.name, p.allergy, p.age);
11.     }
12.     }
13.     catch (……) {
14.     System.out.println("No More Object in file ");
15.     }
16.     catch (Exception e) {
17.     System.out.println("Error while processing file");
18.     }
19.     finally {
20.     ……
21.     }
22.     }
```

answer the following Questions:

The following Exceptions should be added to the method header

- None
- ClassNotFoundExceptin
- FileNotFoundException
- IOException

STION 11

choose the correct answer for **Line 3**:

- FileReader input= new FileReader (fileInput);
- None
- ObjectStream input= new ObjectStream(fileInput);
- DataInputStream input=new DataInputStream(fileInput);

STION 12

choose the correct statement in **line 6**:

- (Patient) input.readObject();
- input.readObject();
- (FPatient) input.readObject();
- (MPatient) input.readObject();

STION 13

choose the correct answer for **Line 13**:

- RuntimeException e
- EOFException e
- ClassNotFoundExceptin e
- None

STION 14

choose the correct answer for **Line 20**:

- input.close();
- break;
- fileInput.close();
- None

Consider the following implementation of **method eligiblePatient()**, this method examines **pList** for eligible patients and write them to Object File *eligiblePatient.dat*

a Patients is eligible if :

- age > =18
- no allergy
- in the case of a female patient no pregnancy

If age < 18 or allergy or pregnancy throw user-defined exception ***IneligiblePatient*** and handle it by printing an appropriate message

```
/* this method examine pList for eligible Patients and write them to Object File
// eligible Patients are patients in age > =18 and no allergy and in case of
// female patient no pregnancy
// if age < 18 or allergy or pregnancy throw exception and handle it by printing
// an appropriate message*/

  1. public void eligiblePatient() throws IOException {
  2.   File F = new File("eligiblePatient.dat");
  3.   ObjectOutputStream Oout = new ObjectOutputStream(F);
  4.   for (int i = 0; i < pList.length; i++) {
  5.    try
  6.    {
  7.      if (pList[i].age < 18)
  8.        throw new IneligiblePatient("age below 18");
  9.      if (pList[i].allergy)
  10.        throw new IneligiblePatient("patient has allergy ");
  11.     if (pList[i] instanceof FPatient && ((FPatient) pList[i]).isPregnent())
  12.      throw new IneligiblePatient("patient is pregnent ");
  13.  ____________________________________
  14.  }
  15.  catch (_____________________) {
  16.  System.out.println(pList[i].name + " ineligible patient " + e.getMessage());}
  17.  catch (_____________________) {
  18.    System.out.println("No More Patient in the List");
  19.    break;
  20.    }
  21.   }
  22. }
```

answer the following questions

Which of the following class need to implement interface **Serializable?**

- Patient
- FPatient
- MPatient
- all

TION 16

The method implementation will compile correctly

- True
- False

TION 17

choose the correct answer for **Line 13**:

- Oout.println(pList[i]);
- Oout.writObject(pList);
- Oout.writ(pList[i]);
- Oout.writObject(pList[i]);

TION 18

choose the correct answer for **Line 15**

- Exception e
- CheckedException e
- None
- IneligiblePatient e

choose the correct answer for **Line 17**

- ArrayIndexOutofBoundException e
- EOFException e
- NullPointerException e
- None

STION 20

if the "*eligiblePatient.dat*" file does not exist, what will happen?

- Will execute properly and produce the output file.
- No exception but will not perform the write operation
- a FileNotFoundException
- None

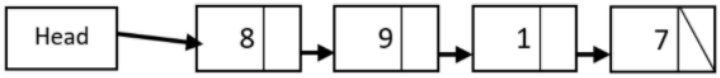
STION 21

if **Oout.close()** is not added at the end of this method, this will

- Will execute properly and produce the output file.
- None
- No exception but will not perform the write operation
- Cause an EOFException

STION 22

Consider the following **Queue**



after **removing 2** elements and **adding 4** and **5**, the resulting Queue is:

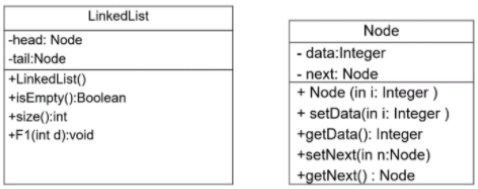
- 8, 9, 4, 5
- 5, 4, 1, 7
- 1, 7, 4, 5
- 4, 5, 8, 9

For an empty **stack s**, choose the correct output for the following sequence of operations.

add 5 to **stack**
add 8 to **stack**
remove from **stack** and **print** the removed item
add 2 to **stack**
add 5 to **stack**
remove from **stack** and **print** the removed item
remove from **stack** and **print** the removed item
remove from **stack** and **print** the removed item
add 1 to **stack**
remove from **stack** and **print** the removed item

- ○ 8 5 2 5 1
- ○ 8 1 2 5 5
- ○ 8 2 5 5 1
- ○ 8 5 5 2 1

Consider the following UML :



If **F1()** has the following implementation:

```
void F1(int d){
      Node current = head, prev = null;
      while (current != null){
         while (current != null && current.getData()!= d){
            prev = current;
            current = current.getNext();
         }
         if (current == null)
            return;

         if (prev != null)
            prev.setNext(current.getNext());
         prev = current;
         current = current.getNext();

      }
}
```

in the main we created myList as follow: 8, 2, 8, 3, 2
After call **myList.F1(2)**
The content of the **mylist** is

- 8, 2, 8, 3, 2
- 8, 2, 8, 2
- 8, 8, 3, 2
- 8, 8, 3

STION 25

if **myList** contains 8, 2, 8, 3, 2 After call **myList.F1(8)**
The content of the **mylist** is

- 8, 2, 3, 2
- 2, 3, 2
- 8, 2, 8, 3, 2
- 2, 8, 3, 2

can you generalize what does **F1** do

- delete all occurrence of d in the list except the head
- delete the first occurrence of d in the list
- delete all occurrence of d in the list
- nothing change

ITION 27

Complete the method ***removeBetween*** in class "***LinkedList***", the list holds characters data. the method receives two characters and removes the nodes between them if both characters are in the list.

Example 1 :
if the list
A → B → C → D → E
after calling the method ***removeBetween('B','E')***, the result list is :
A → B → E
Example 2 :
if the list
A → F → B → D
after calling the method ***removeBetween('B','E')*** , the result list is :
A → F → B → D

```
public void removeBetween (char i , char j){
1. if(isEmpty())
2.   return;
3.     Node p = head , q = head;
4.     ..........................................
5.     ..........................................
6.     ..........................................
7.     ..........................................
8.     ..........................................
9.     ..........................................
10.    ..........................................
11.  }
```

choose the correct answer for **Line 4**

- while (p != null && p.getData() == i)
- while (p < null && p.getData()!= i)
- while (p == null && p.getData() != i)
- while (p != null && p.getData() != i)

ITION 28

choose correct answer for **Line 5**

- p=head;
- p++;
- p=q;
- p=p.getNext();

ITION 29

choose correct answer for **Line 6**

- if(p==tail) break;
- if(p!=null) return;
- if(p==null) return;
- if(p==null) break;

choose the correct answer for **Line 7**

- while (q != null && q.getData() == j)
- while (q != null && q.getData() != j)
- while (q < null && q.getData() != j)
- while (q == null && q.getData() != j)

STION 31

choose the correct answer for **Line 8**

- q=p;
- q++;
- q=head;
- q=q.getNext();

STION 32

choose the correct answer for **Line 9**

- if(q==null) return;
- if(q!=null) return;
- if(q==null) break;
- if(q==tail) break;

STION 33

choose the correct answer for **Line 10**

- q=p;
- q.setNext(p);
- p=q;
- p.setNext(q);

what does this code do? choose the most suitable answer

```
public void f ()
    {
        Node p = head;
        head = head.getNext();
        p.setNext(null);
        Node q=head;
        while(q.getNext()!=null)
            q= q.getNext();
        q.setNext(p);
}
```

- insert **p** at the end of the list
- remove **p** from list
- cause a null pointer exception
- insert **q** after **p**

ESTION 35

To design a recursion method, which of the following is needed:

- None
- At least one base case and up to two general cases.
- At most one base case and one general case.
- At least one base case, and one general case.

ESTION 36

Recursion is a special type of :

- Branch control.
- Repetition control.
- Conditional control.
- None

For the following method, determine the base case and the general cases for the recursion call:

Calculate the power m for the number n.

```
public static int power (int n, int m){
if ..........
   return n;
else
   ..........
}
```

Base case is:

- m==0
- None
- m==1 || m==0
- m==1

ESTION 38

the general case is :

- return 1;
- return n*power(n, m-1);
- return n*power(n-1,m);
- return ;

ESTION 39

Given the following method declaration,

```
public static boolean p (int [] a , int i , int f)
{
if (i<f){
   if (a[i] == a[f]){
      return p(a,i+1,f-1);}
   else{
      return false;}
   }
else{
   return true;}
}
```

and if **`a = int [1,2,3,2,1]`**. suppose a calling to the method as follow **`p(a,0,4)`**.
The Recursion Method will be called (including the main call):

- three times then it will cause a run time error
- five times
- once then it will cause a run time error
- four times

at the third call, the value of **i** and **f** will be:

- i=2 and f=1
- i=0 and f=4
- i=2 and f=2
- i=2 and f=3

ESTION 41

and the returned value will be :

- No value , a run time error
- true
- false
- No value , compilation error

ESTION 42

What is the output of the following method, if **n = 5**

```
public static int sum (int num)
{
   if (num == 0)
       return 0;
  else
      return num + sum (num);
}
```

- 15
- cause compilation error
- cause a run time error
- 10

Given the following method :

```
static void printPattern(int n, int m, boolean flag)
    {
        System.out.print(m + " ");
        if (flag == false && n == m)
            return;
        if (flag) {
            if (m - 5 > 0)
                printPattern(n, m - 5, true);
            else
                printPattern(n, m - 5, false);
        }
        else
            printPattern(n, m + 5, false);
    }
```

what is the output for the following call **printPattern(0,5,true)**

- 5 0
- 0 5
- 5 10
- 0 10