Scheme of Evaluation

Internal Assessment Test 3 – July 2022

Sub:	Programming in Java							**Code:**	18CS653
Date:	12/7/2022	Duration:	90mins	Max Marks:	50	**Sem:**	VI	**Branch:**	EEE/MECH

Note: Answer Any five full questions.

Question #		**Description**	**Marks Distribution**		**Max Marks**
1	a)	What is package? Write a simple program to create a package and import the package to make use of a class declared in a package Definition of package Define package Use package	1M 2M 3M	6M	10M
	b)	Explain the Access Protection in Java Public Private Protected default	1M 1M 1M 1M	4M	
2	a)	What is an interface? Design an interface called polygon with a method called area. Implement this interface to create different classes like square, rectangle and print area of square and rectangle Interface definition Define interface Class Rectangle Class Square Print area	2M 2M 2M 2M 2M	10M	10M
3	a)	What is an abstract class? Illustrate how we can make use of abstract class and concrete methods written in an abstract class with an example Definition abstract class Define abstract class Abstract method	2M 2M 1M 1M 2M 2M	10M	10M

		Concrete method Use abstract class output			
4	a)	Demonstrate the working of nested try catch block with an example Try block Catch block	3M 3M	6M	10M
4	b)	Define an exception. Write a simple program to generate arithmetic exception and print the description of exception through the program Definition of exception example	1M 3M	4M	
5	a)	What is String in Java? Explain any 4 String constructors with an example String definition Any four constructors with example	2M 2M*4	10M	10M
6	a)	Explain how to modify the string using following methods i)substring( ) ii) concat( ) iii) replace( ) iv) trim( ) substring concat replace trim	2.5M *4	10M	10M

Q. 1 a) What is package? Write a simple program to create a package and import the package to make use of a class declared in a package

Packages

A package as the name suggests is a pack(group) of classes, interfaces and other packages. In java we use packages to organize our classes and interfaces. We have two **types of packages in Java**: built-in packages and the packages we can create (also known as user defined package). In this guide we will learn what are packages, what are user-defined packages in java and how to use them.

In java we have several built-in packages, for example when we need user input, we import a package like this:

import java.util.Scanner

Here:
→ **java** is a top level package
→ **util** is a sub package
→ and **Scanner** is a class which is present in the sub package **util**.

```
//save by A.java
package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}


//save by B.java
import pack.A;

class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

```
Output:Hello
```

To Compile:

```
javac -d . A.java
javac B.java
```

To Run: java B

Q. 1 b) Explain the Access Protection in Java

There are four types of Java access modifiers:

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

Let's understand the access modifiers in Java by a simple table.

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Q. 2 a) What is an interface? Design an interface called polygon with a method called area. Implement this interface to create different classes like square, rectangle and print area of square and rectangle

An **interface in java** is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is *a mechanism to achieve abstraction*. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Java Interface also **represents the IS-A relationship**.

It cannot be instantiated just like the abstract class.

```
interface polygon
{
double area ( );
}


class Rectangle implements  polygon
{
double l,b;
void getvalues( )//you can take values from user as well
{
l = 10.5F
b = 7.3F
}
public double area ( )
{
return (l*b);
}
}


Class Square extends Rectangle implements polygon
{
public double area( )
}
return (l*b);
}
}
```

```
class prog6b
{
public static void main( String [ ] args )
{
Rectangle R = new Rectangle ( );
R.getvalues( );
System.out.println("Area of rectangle =" +R.area( ));
Square S = new Square( );
S.getvalues( );
System.out.println("Area of square=" + S.area( ));
}
}
```

Output:

```
Area of rectangle = 76.65000200271606
Area of square = 76.65000200271606
```

Q. 3 a) What is an abstract class? Illustrate how we can make use of abstract class and concrete methods written in an abstract class with an example.

Abstract class in Java

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

```
// A Simple demonstration of abstract.
abstract class A {
abstract void callme();

// concrete methods are still allowed in abstract classes
void callmetoo() {
System.out.println("This is a concrete method.");
}
}

class B extends A {
void callme() {
System.out.println("B's implementation of callme.");
}
}

class AbstractDemo {
public static void main(String args[]) {
B b = new B();
b.callme();
b.callmetoo();
}
}
```

Output:

```
B's implementation of callme
This is a concrete method
```

Q. 4 a) Demonstrate the working of nested try catch block with an example.

What is an exception?

An Exception is an unwanted event that interrupts the normal flow of the program. When an exception occurs, the program execution gets terminated. In such cases we get a system generated error message. The good thing about exceptions is that they can be handled in Java. By handling the exceptions we can provide a meaningful message to the user about the issue rather than a system generated message, which may not be understandable to a user.

Java Nested try block

In Java, using a try block inside another try block is permitted. It is called as nested try block. Every statement that we enter a statement in try block, context of that exception is pushed onto the stack.

For example, the **inner try block** can be used to handle **ArrayIndexOutOfBoundsException** while the **outer try block** can handle the **ArithemeticException** (division by zero).

```
class Excep6{

 public static void main(String args[]){

  try{

    try{

     System.out.println("going to divide");

     int b =39/0;

    }catch(ArithmeticException e){System.out.println(e);}
    try{

    int a[]=new int[5];

    a[5]=4;

    }catch(ArrayIndexOutOfBoundsException e){System.out.println(e);}


    System.out.println("other statement);

   }catch(Exception e){System.out.println("handeled");}


   System.out.println("normal flow..");

  }

 }
```

Output:

Q. 4 b) Define an exception. Write a simple program to generate arithmetic exception and print the description of exception through the program

What is an exception?

An Exception is an unwanted event that interrupts the normal flow of the program. When an exception occurs, the program execution gets terminated. In such cases we get a system generated error message. The good thing about exceptions is that they can be handled in Java. By handling the exceptions we can provide a meaningful message to the user about the issue rather than a system generated message, which may not be understandable to a user.

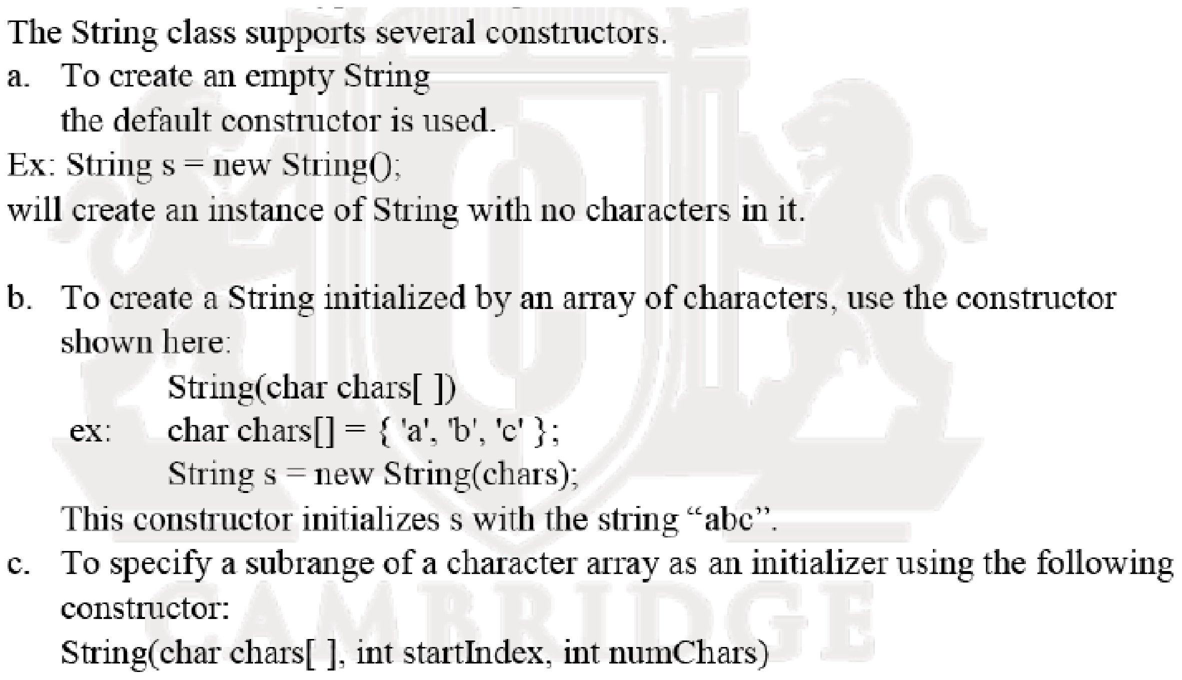
```
class Exc2 {
public static void main(String args[])
 {
        int d, a;
        try { // monitor a block of code.
        d = 0;
        a = 42 / d;
        System.out.println("This will not be printed.");
} catch (ArithmeticException e) {
System.out.println("Exception: " + e);
}
}
```

Output: `Exception: java.lang.ArithmeticException: / by zero`

Q. 5 a) What is String in Java? Explain any 4 String constructors with an example

Strings in Java

Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character '\0'.



The String class supports several constructors.

a. To create an empty String
   the default constructor is used.

Ex: String s = new String();

will create an instance of String with no characters in it.

b. To create a String initialized by an array of characters, use the constructor shown here:

```
        String(char chars[ ])
 ex:    char chars[] = { 'a', 'b', 'c' };
        String s = new String(chars);
```

   This constructor initializes s with the string "abc".

c. To specify a subrange of a character array as an initializer using the following constructor:

   String(char chars[ ], int startIndex, int numChars)

Here, startIndex specifies the index at which the subrange begins, and numChars specifies the number of characters to use. Here is an example:

```
char chars[] = { 'a', 'b', 'c', 'd', 'e', 'f' };
String s = new String(chars, 2, 3);
```

This initializes s with the characters cde.

d. To construct a String object that contains the same character sequence as another String object using this constructor:

    String(String strObj)

    Here, strObj is a String object.

```
class MakeString
{ public static void main(String args[])
{ char c[] = {'J', 'a', 'v', 'a'};
String s1 = new String(c);


 String s2 = new String(s1);
System.out.println(s1);
 System.out.println(s2);
}
}
```

The output from this program is as follows:

```
Java
Java
```

As you can see, s1 and s2 contain the same string.

e. To Construct string using byte array:

Even though Java's char type uses 16 bits to represent the basic Unicode character set, the typical format for strings on the Internet uses arrays of 8-bit bytes constructed from the ASCII character set.

Because 8-bit ASCII strings are common, the String class provides constructors that initialize a string when given a byte array.

Ex: String(byte asciiChars[ ])

String(byte asciiChars[ ], int startIndex, int numChars)

The following program illustrates these constructors:

```
class SubStringCons
{ public static void main(String args[])
{
byte ascii[] = {65, 66, 67, 68, 69, 70 };
String s1 = new String(ascii);
System.out.println(s1);
String s2 = new String(ascii, 2, 3);
System.out.println(s2);
}
}
```

This program generates the following output:

```
ABCDEF
CDE
```

f. To construct a String from a StringBuffer by using the constructor shown here:

Ex: String(StringBuffer strBufObj)

g. Constructing string using Unicode character set and is shown here:

Q. 6 a) Explain how to modify the string using following methods
i)substring( ) ii) concat( ) iii) replace( ) iv) trim( )

i) substring( )- A part of string is called **substring**. In other words, substring is a subset of another string. In case of substring startIndex is inclusive and endIndex is exclusive.

```
public class TestSubstring{

 public static void main(String args[]){

 String s="SachinTendulkar";

  System.out.println(s.substring(6));//Tendulkar

  System.out.println(s.substring(0,6));//Sachin

 }

}
```

Output:

```
Tendulkar
Sachin
```

ii) concat( ) - string concatenation forms a new string *that is* the combination of multiple strings.

```
class TestStringConcatenation3{

 public static void main(String args[]){

 String s1="Sachin ";

 String s2="Tendulkar";

 String s3=s1.concat(s2);

 System.out.println(s3);//Sachin Tendulkar

 }

}
```

Output: Sachin Tendulkar

iii)replace( )

The **java string replace**() method returns a string replacing all the old char or CharSequence to new char or CharSequence.

```
public class ReplaceExample1{

public static void main(String args[]){

String s1="javatpoint is a very good website";

String replaceString=s1.replace('a','e');//replaces all occurrences of 'a' to 'e'

System.out.println(replaceString);

}

}
```

Output: jevetpoint is e very good website

iv) trim( )

The **java string trim**() method eliminates leading and trailing spaces.

```
public class StringTrimExample{

public static void main(String args[]){

String s1="  hello string   ";

System.out.println(s1+"javatpoint");//without trim()

System.out.println(s1.trim()+"javatpoint");//with trim()

}

}
```

Output:

```
hello string   javatpoint
hello stringjavatpoint
```