

1. Write a Java Program to demonstrate the creation of class for student information.

```
package sss;
import java.util.*;

public class student {
    public String vtuno;
    public String fullname;
    public String sem;
    public String branch;
    public String address;

    public student() {
        Scanner s = new Scanner(System.in);
        System.out.println("VTU NO : ");
        vtuno = s.nextLine();
        System.out.println("Full name : ");
        fullname = s.nextLine();
        System.out.println("Address : ");
        address = s.nextLine();
        System.out.println("Branch : ");
        branch = s.nextLine();
        System.out.println("Semester : ");
        sem = s.nextLine();
    }

    public void show() {
        System.out.println("Entered data");
        System.out.println("VTU NO : "+vtuno);
        System.out.println("Full name : "+fullname);
        System.out.println("Semester : "+sem);
        System.out.println("Branch : "+branch);
        System.out.println("Address : "+address);
    }

    public static void main(String[] args) {
        student std = new student();
        std.show();
    }
}
```



```
<terminated> student [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 4:07:20 AM - 4:07:47 AM) [pid: 14060]
VTU NO :
3PD21000
Full name :
ABC PQR
Address :
Karnataka
Branch :
CSE
Semester :
4
Entered data
VTU NO : 3PD21000
Full name : ABC PQR
Semester : 4
Branch : CSE
Address : Karnataka
```

Viva Questions :

1. What is the purpose of a constructor in a Java class, as demonstrated in the `student` class?

The purpose of a constructor in a Java class is to initialize the instance variables of an object when that object is created. In the `student` class, the constructor gathers user input and assigns it to the instance variables, creating a `student` object with specific data.

2. How do you create an object of a class in Java?

You create an object of a class in Java using the `new` keyword, followed by the class constructor. For example, in the `main` method, `student std = new student();` creates a `student` object named `std`.

3. What is the significance of the `public` keyword before the class and instance variable declarations?

The `public` keyword indicates that the class and instance variables are accessible from any other class. In this case, it allows the `main` method to create an instance of the `student` class and access its members.

4. How are instance variables different from local variables?

Instance variables (also known as fields) are variables declared within a class but outside of any method. They belong to an object of the class and have class-wide scope. Local variables are declared within a method and have method-level scope, meaning they are only accessible within that method.

5. What is the purpose of the `show()` method in the `student` class?

The `show()` method is used to display the student information stored in the instance variables of a `student` object. It is a convenient way to access and display the data associated with an object.

6. What is the role of the `Scanner` class in this program, and how does it work?

The `Scanner` class is used to read user input from the console. It works by creating a `Scanner` object, and then you can use its methods (e.g., `nextLine()`) to read different types of input (strings, numbers, etc.) from the keyboard.

Experiment 2:

Write a program in Java for String handling which performs the following:

- i) Checks the capacity of String Buffer objects.
- ii) Reverses the contents of a string given on console and converts the resultant string in upper case.
- iii) Reads a string from console and appends it to the resultant string of ii.

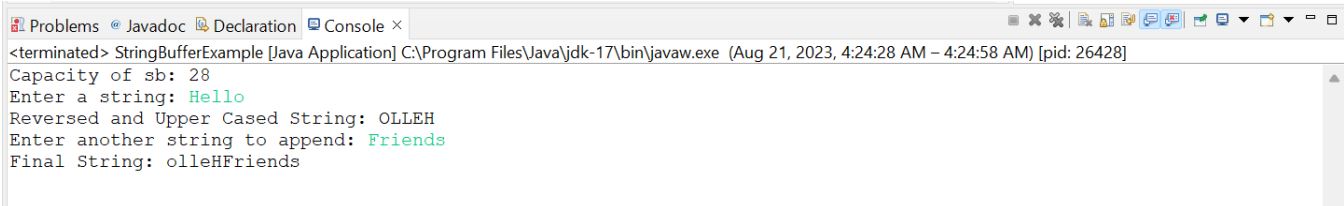
```
package sss;
import java.util.Scanner;

public class StringBufferExample {
    public static void main(String[] args) {

        // Checking capacity of StringBuffer object
        StringBuffer sb = new StringBuffer("Hello World!");
        System.out.println("Capacity of sb: " + sb.capacity());

        // Reversing the string and converting to upper case
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = sc.nextLine();
        StringBuffer sb1 = new StringBuffer(input);
        sb1.reverse(); // reversing the string
        String upperCase = sb1.toString().toUpperCase(); // converting to upper case
        System.out.println("Reversed and Upper Cased String: " + upperCase);

        // Appending a string to the reversed and upper cased string
        System.out.print("Enter another string to append: ");
        String appendStr = sc.nextLine();
        sb1.append(appendStr); // appending the string
        System.out.println("Final String: " + sb1.toString());
    }
}
```



The screenshot shows the console output of the Java program. The output is as follows:

```
<terminated> StringBufferExample [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 4:24:28 AM - 4:24:58 AM) [pid: 26428]
Capacity of sb: 28
Enter a string: Hello
Reversed and Upper Cased String: OLLEH
Enter another string to append: Friends
Final String: olleHFriends
```

Viva Questions :

1. What is the purpose of the program for String handling?

The purpose of this program is to demonstrate various operations on strings using `StringBuffer` and `StringBuilder` classes. It checks the capacity of `StringBuffer` objects, reverses the contents of a string, converts it to uppercase, and appends a new string to the result.

2. What is the difference between `String` and `StringBuffer` in Java?

`String` objects in Java are immutable, meaning their values cannot be changed after creation. In contrast, `StringBuffer` objects are mutable, and you can modify their content without creating new objects.

3. How do you check the capacity of a `StringBuffer` object?

You can check the capacity of a `StringBuffer` object using the `capacity()` method. For example, `int capacity = stringBuffer.capacity();` will store the capacity of the `StringBuffer` object in the `capacity` variable.

4. How is the `reverse()` method used to reverse the contents of a string?

The `reverse()` method is invoked on a `StringBuffer` or `StringBuilder` object to reverse its contents. For example, `stringBuffer.reverse();` will reverse the characters in `stringBuffer`.

5. How do you convert a string to uppercase in Java?

You can convert a string to uppercase in Java by using the `toUpperCase()` method of the `String` class. For example, `String upperCaseString = inputString.toUpperCase();` will store the uppercase version of `inputString` in `upperCaseString`.

6. How can you append a new string to an existing string in Java?

You can append a new string to an existing string in Java using the `append()` method of a `StringBuffer` or `StringBuilder` object. For example, `stringBuffer.append(newString);` will append the contents of `newString` to the `stringBuffer`.

7. What is the significance of using `StringBuffer` or `StringBuilder` over regular strings (`String`) for concatenation operations?

****Answer:**** `StringBuffer` and `StringBuilder` are more efficient for concatenation operations because they are mutable, and modifications can be done in-place. In contrast, with regular strings (`String`), every concatenation creates a new string object, which can be inefficient for large operations.

8. What is the difference between `StringBuffer` and `StringBuilder`?

Both `StringBuffer` and `StringBuilder` are mutable, but `StringBuffer` is synchronized, making it thread-safe but potentially slower in a multi-threaded environment. `StringBuilder` is not synchronized and is faster for single-threaded operations.

3 a. Write a JAVA Program to demonstrate Constructor Overloading and Method Overloading.

Constructor Overloading

```
package sss;
class MyClass {
    int x;
    MyClass() {
        System.out.println("Inside MyClass().");
        x=0;
    }

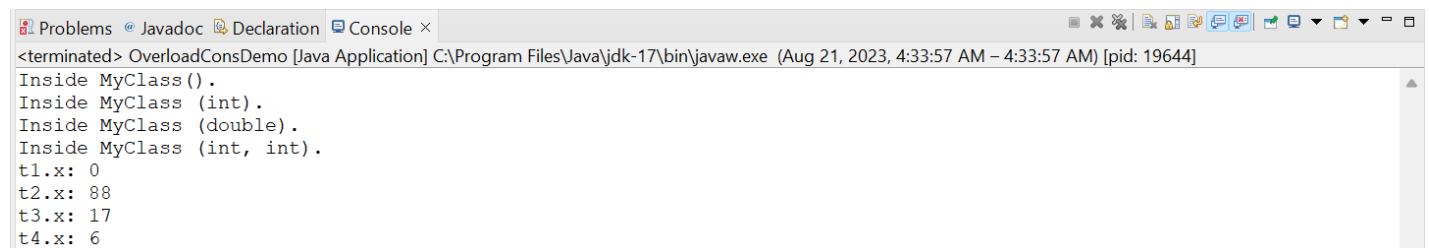
    MyClass (int i ) {
        System.out.println("Inside MyClass (int).");
        x=i;
    }

    MyClass (double d) {
        System.out.println("Inside MyClass (double).");
        x= (int) d;
    }

    MyClass (int i, int j) {
        System.out.println("Inside MyClass (int, int).");
        x=i+j;
    }
}

public class OverloadConsDemo {
    public static void main(String[] args) {
        MyClass t1 = new MyClass();
        MyClass t2 = new MyClass(88);
        MyClass t3 = new MyClass(17.23);
        MyClass t4 = new MyClass(2, 4);

        System.out.println("t1.x: " + t1.x);
        System.out.println("t2.x: " +t2.x);
        System.out.println("t3.x: " +t3.x);
        System.out.println("t4.x: " +t4.x);
    }
}
```

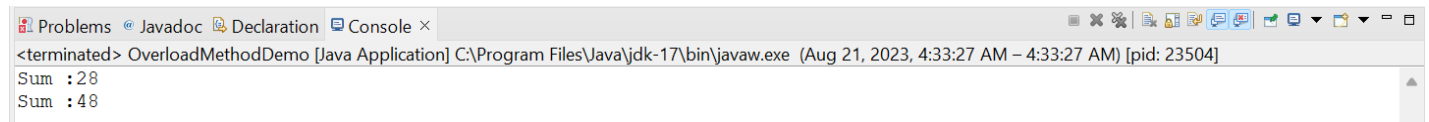


```
<terminated> OverloadConsDemo [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 4:33:57 AM - 4:33:57 AM) [pid: 19644]
Inside MyClass().
Inside MyClass (int).
Inside MyClass (double).
Inside MyClass (int, int).
t1.x: 0
t2.x: 88
t3.x: 17
t4.x: 6
```

Method Overloading

```
package sss;
public class OverloadMethodDemo {
    public static void main(String[] args) {
        Addn s = new Addn();
        s.Sum(12,16);
        s.Sum(12,16,20);
    }
}

class Addn{
    void Sum(int a, int b)
    {
        System.out.println("Sum :" + (a+b));
    }
    void Sum(int a, int b, int c)
    {
        System.out.println("Sum :" + (a+b+c));
    }
}
```



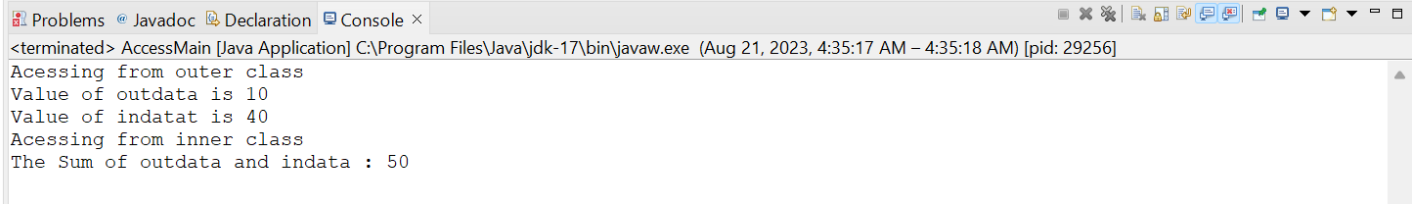
The screenshot shows an IDE interface with a 'Console' tab selected. The console output displays the results of the program execution: 'Sum :28' followed by 'Sum :48'. The window title bar indicates the application is 'OverloadMethodDemo [Java Application]' running on 'C:\Program Files\Java\jdk-17\bin\javaw.exe' with PID 23504.

```
<terminated> OverloadMethodDemo [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 4:33:27 AM - 4:33:27 AM) [pid: 23504]
Sum :28
Sum :48
```

b. Write a JAVA Program to implement Inner class and demonstrate its Access Protections.

```
package sss;
import java.io.*;
class outer {
    int outdata = 10;
    void Display() {
        inner inobj = new inner();
        System.out.println("Acessing from outer class ");
        System.out.println ("Value of outdata is "+outdata);
        System.out.println("Value of indatat is "+inobj.indata);
    }
    class inner {
        int indata = 40;
        void inmethod() {
            System.out.println("Acessing from inner class");
            System.out.println("The Sum of outdata and indata : " +(outdata + indata));
        }
    }
}
public class AccessMain {

    public static void main(String[] args) {
        outer outobj = new outer();
        outobj.Display();
        outer.inner inobj1 = outobj.new inner();
        inobj1.inmethod();
    }
}
```



The screenshot shows a Java IDE window with a console tab. The console output is as follows:

```
<terminated> AccessMain [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 4:35:17 AM – 4:35:18 AM) [pid: 29256]
Acessing from outer class
Value of outdata is 10
Value of indatat is 40
Acessing from inner class
The Sum of outdata and indata : 50
```

Viva Questions :

****Question 1:**** What is the purpose of Constructor Overloading and Method Overloading in Java?

****Answer:**** Constructor Overloading allows a class to have multiple constructors with different parameters, providing flexibility when creating objects. Method Overloading enables a class to have multiple methods with the same name but different parameter lists, simplifying method invocation with various arguments.

****Question 2:**** How does Constructor Overloading work in Java, and what is the advantage?

****Answer:**** Constructor Overloading allows a class to have multiple constructors with different parameter lists. The advantage is that it provides flexibility when creating objects, allowing different combinations of arguments to initialize objects.

****Question 4:**** How does Method Overloading work in Java, and what is the advantage?

****Answer:**** Method Overloading allows a class to have multiple methods with the same name but different parameter lists. The advantage is that it simplifies method invocation by allowing the same method name to be used with different argument combinations.

****Question 6:**** What is an Inner Class in Java?

****Answer:**** An Inner Class in Java is a class defined within another class. It has access to the members of the enclosing class, including private members. Inner classes are used for better encapsulation and organization of code.

****Question 7:**** What are the access protections available for Inner Classes in Java?

****Answer:**** Inner Classes in Java can have the following access protections:

- `public`: The inner class can be accessed from any other class.
- `protected`: The inner class can be accessed from the same package or subclasses.
- `default` (package-private): The inner class can be accessed only within the same package.
- `private`: The inner class can be accessed only from the enclosing class.

****Question 9:**** How do you create an object of an Inner Class in Java?

****Answer:**** To create an object of an Inner Class, you first create an object of the Outer Class, and then use it to create an object of the Inner Class. For example:

```
Outer outerObject = new Outer();
```

```
Outer.Inner innerObject = outerObject.new Inner();
```


4. a. Write and execute a JAVA Program to demonstrate Inheritance.(single level and multilevel)

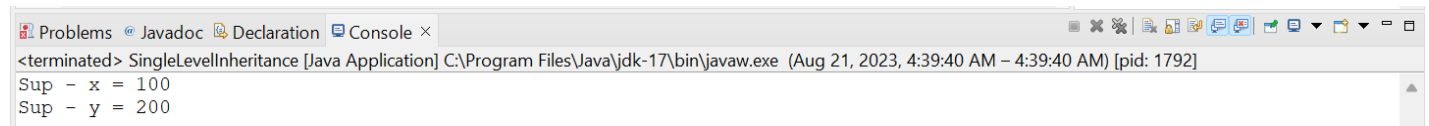
single level

```
package sss;
class sup {
    int x;
    sup(int x) {
        this.x = x;
    }
    void display() {
        System.out.println("Sup - x = "+x);
    }
}

class supr extends sup
{
    int y;
    supr(int x, int y) {
        super(x);
        this.y = y;
    }
    void display(){
        System.out.println("Sup - x = "+x);
        System.out.println("Sup - y = "+y);
    }
}

public class SingleLevelInheritance {

    public static void main(String[] args) {
        supr s1 = new supr(100, 200);
        s1.display();
    }
}
```



```
Problems @ Javadoc Declaration Console x
<terminated> SingleLevelInheritance [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 4:39:40 AM - 4:39:40 AM) [pid: 1792]
Sup - x = 100
Sup - y = 200
```

Multilevel

```
package sss;
class length {
    int l ;
    length(int l){
        this.l=l;
    }
}
class breadth extends length {
    int b;
    breadth(int l, int b) {
        super(l);
        this.b = b;
    }
}

class volume extends breadth {
    int h;
    volume(int l, int b , int h) {
        super(l,b);
        this.h = h;
    }
    void display() {
        System.out.println("Volume : "+(l*b*h));
    }
}

public class MultiLevelInheritance {

    public static void main(String[] args) {
        volume s1 = new volume(100, 200, 300);
        s1.display();
    }
}
```

Problems @ Javadoc Declaration Console ×

<terminated> MultiLevelInheritance [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 4:40:46 AM – 4:40:47 AM) [pid: 26528]

Volume : 6000000

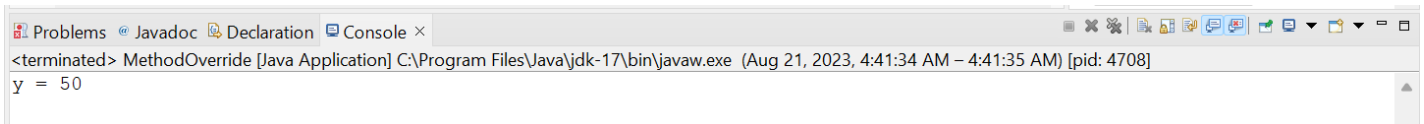
b. Write and execute a JAVA program to demonstrate method overriding.

```
package sss;
class A {
    int x;
    A(int x) {
        this.x = x;
    }
    void Show() {
        System.out.println("x = "+x);
    }
}

class B extends A
{
    int y;
    B(int x, int y) {
        super(x);
        this.y = y;
    }
    void Show() {
        System.out.println("y = "+y);
    }
}

public class MethodOverride {

    public static void main(String[] args) {
        B s1 = new B(45,50);
        s1.Show();
    }
}
```



The screenshot shows an IDE window with a console tab. The console title is "Console x". The output text is "<terminated> MethodOverride [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 4:41:34 AM – 4:41:35 AM) [pid: 4708]" followed by "y = 50".

Viva Questions :

****Question 1:**** What is Inheritance in Java, and why is it used?

****Answer:**** Inheritance in Java is a mechanism by which a class inherits the properties and behaviors (fields and methods) of another class. It is used to promote code reuse and establish relationships between classes.

****Question 2:**** What is Single-Level Inheritance in Java?

****Answer:**** Single-Level Inheritance occurs when a class inherits properties and behaviors from a single superclass (parent class). It represents a simple one-level inheritance hierarchy.

****Question 4:**** What is Multi-Level Inheritance in Java?

****Answer:**** Multi-Level Inheritance occurs when a class inherits properties and behaviors from a superclass, which, in turn, inherits from another superclass. It forms a chain of inheritance.

****Question 6:**** What is Method Overriding in Java?

****Answer:**** Method Overriding is a feature that allows a subclass to provide a specific implementation of a method that is already defined in its superclass. The method in the subclass has the same name, return type, and parameters as the method in the superclass.

****Question 9:**** What happens when a subclass overrides a method from its superclass?

****Answer:**** When a subclass overrides a method from its superclass, the version of the method in the subclass is called instead of the superclass's version when the method is invoked on an object of the subclass.

****Question 10:**** What are the advantages of using Inheritance and Method Overriding in Java?

****Answer:**** The advantages of using Inheritance and Method Overriding include code reuse, improved organization of code, and the ability to define specific behaviors in subclasses while maintaining a consistent interface in the superclass. This promotes modularity and maintainability in large-scale applications.

5. Write a JAVA program which has

- i) A Class called Account that creates account with 500Rs minimum balance, a deposit() method to deposit amount, a withdraw() method to withdraw amount and also throws Less Balance Exception if an account holder tries to withdraw money which makes the balance become less than 500Rs.
- ii) A Class called Less Balance Exception which returns the statement that says withdraw amount (___Rs) is not valid.

```
package sss;
import java.io.*;
class Less_Balance_Exception extends Exception {
    String msg = " ";
    int bal;
    Less_Balance_Exception(int amount) {
        msg = "Less Balance";
        bal = amount;
    }
}

class Account {
    int bal;
    Account() {
        bal = 500;
    }
    Account (int minbal) {
        bal = minbal;
    }
    void Withdraw(int amount) throws Less_Balance_Exception {
        int newbal;
        newbal = bal;
        newbal -= amount;
        if(newbal < 500)
            throw new Less_Balance_Exception(amount);
        bal = newbal;
        System.out.println("Amount withdrawn" +amount);
    }
    void Deposit(int amount) {
        bal += amount;
    }
    int GetBal(){
        return bal;
    }
}

public class ExceptionHandling {

    public static void main(String[] args)
        throws IOException {
        Account a1 = new Account();
        System.out.println("Balance of account " +a1.GetBal());
        System.out.println("Enter amount to be deposited");
        DataInputStream dis = new DataInputStream(System.in);
        int amt = Integer.parseInt(dis.readLine());
        a1.Deposit(amt);
        System.out.println("Amount " +amt +" Deposited");

        try {
            System.out.println("Enter withdrawl Amount");
```

```

        int w = Integer.parseInt(dis.readLine());
        a1.Withdraw(w);
    }
    catch (Less_Balance_Exception lb) {
        System.out.println("Amount " + lb.bal + " cannot be withdrawn");
    }
}
}

```

```

<terminated> ExceptionHandling [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 4:49:28 AM - 4:49:36 AM) [pid: 28460]
Balance of account 500
Enter amount to be deposited
1500
Amount 1500 Deposited
Enter withdrawl Amount
1900
Amount 1900 cannot be withdrawn

```

iii) A Class which creates 2 accounts, both account deposit money and one account tries to withdraw more money which generates a Less Balance Exception take appropriate action for the same.

```

package sss;
import java.io.*;
class lessbalanceexception extends Exception
{
    String msg = " ";
    int bal;
    lessbalanceexception (int amount)
    {
        msg = "less balance";
        bal = amount;
    }
}
class Accounts{
    int bal;
    Accounts()
    {
        bal = 500;
    }
    Accounts(int minbal)
    {
        bal = minbal;
    }
    void withdraw(int amount) throws lessbalanceexception
    {
        int newbal;
        newbal = bal;
        newbal -= amount;
        if ( newbal < 500)
            throw new lessbalanceexception(amount);
        bal = newbal;
        System.out.println("Amount withdrawn : "+amount);
    }
    void deposit(int amount)
    {
        bal+=amount;
    }
    int getbal()

```

```

    {
        return bal;
    }
}

public class ExceptionHandling_2 {
    public static void main(String args[])
        throws IOException {
        Accounts a1 = new Accounts();
        Accounts a2 = new Accounts();
        System.out.println("Two Accounts created with minimum balance\n");
        System.out.println("Balance of Account 1 : "+a1.getbal()+"\n");
        System.out.println("Balance of Account 2 : "+a2.getbal()+"\n");
        System.out.println("Enter amount to be deposited in Account 1 : ");
        DataInputStream dis = new DataInputStream(System.in);
        int amt1 = Integer.parseInt(dis.readLine());
        a1.deposit(amt1);
        System.out.println("\namount "+amt1+" deposited in Account 1\n");
        System.out.println("Enter amount to be deposited in Account 2 : ");
        int amt2 = Integer.parseInt(dis.readLine());
        a2.deposit(amt2);
        System.out.println("\namount "+amt2+" deposited in Account 2\n");
        try{
            System.out.println("Enter withdrawl amount from Account 1 : ");
            int w = Integer.parseInt(dis.readLine());
            a1.withdraw(w);
        }
        catch(lessbalanceexception lb)
        {
            System.out.println("\nAmount " + lb.bal + " cannot be withdrawn");
        }
    }
}

```

```

Problems @ Javadoc Declaration Console x
<terminated> ExceptionHandling_2 [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 4:59:37 AM - 4:59:55 AM) [pid: 2188]
Two Accounts created with minimum balance

Balance of Account 1 : 500

Balance of Account 2 : 500

Enter amount to be deposited in Account 1 :
1000
amount 1000 deposited in Account 1

Enter amount to be deposited in Account 2 :
1300
amount 1300 deposited in Account 2

Enter withdrawl amount from Account 1 :
1400
Amount 1400 cannot be withdrawn

```

Viva Questions

****Question 1:**** What is an exception in Java, and how does it differ from a regular program flow?

****Answer:**** An exception in Java is an event that occurs during the execution of a program and disrupts the normal flow of instructions. It differs from regular program flow because it can interrupt the program's execution and requires special handling.

****Question 2:**** What is the purpose of exception handling in Java?

****Answer:**** The purpose of exception handling in Java is to gracefully handle and recover from unexpected or erroneous situations that may occur during program execution, ensuring that the program does not crash and can continue to run in a controlled manner.

****Question 4:**** What is the difference between the `try`, `catch`, and `finally` blocks in Java exception handling?

****Answer:****

- The `try` block contains the code that may throw an exception.
- The `catch` block is used to handle the exception if it occurs. It specifies what actions should be taken when a specific exception is thrown.
- The `finally` block contains code that is executed regardless of whether an exception is thrown or not. It is often used for cleanup operations.

****Question 5:**** How can you create a custom exception class in Java?

****Answer:**** To create a custom exception class in Java, you typically define a class that extends `Exception` or one of its subclasses. You can add constructors and methods to customize the exception's behavior and error messages.

****Question 6:**** What is the purpose of the `throw` statement in Java?

****Answer:**** The `throw` statement is used to explicitly throw an exception within your code. It is used to signal that an exceptional condition has occurred and that the program should transfer control to the nearest `catch` block that can handle that exception.

****Question 8:**** What is the purpose of the `throws` keyword in method declarations?

****Answer:**** The `throws` keyword in method declarations is used to indicate that a method may throw certain exceptions, but it does not handle them within the method. It allows the calling code to be aware of the exceptions that need to be handled or propagated.

Certainly, here are some more generalized Java-related viva questions and their answers related to exception handling:

****Question 13:**** What is the role of the `RuntimeException` class in Java exception hierarchy?

****Answer:**** `RuntimeException` is the superclass of all unchecked exceptions in Java. Unchecked exceptions are not required to be caught or declared, and they typically represent programming errors or exceptional conditions that cannot be anticipated at compile time.

6. Write a Java program to implement multithreading in JAVA which demonstrate built in methods available for thread.

```
package sss;
import java.lang.Thread;

public class ThreadDemo extends Thread {
    public void run() {
        System.out.println(Thread.currentThread().getName()+" is running...");
        System.out.println("Priority is : " +Thread.currentThread().getPriority());
        System.out.println("Name is : " +Thread.currentThread().getName());
        for(int j=1;j<=5; j++){
            System.out.println("User "+Thread.currentThread().getName());
            try {
                Thread.sleep(1000);
            }
            catch (InterruptedException e) {
                System.out.println("Exception has occurred" + e);
            }
        }
        System.out.println(Thread.currentThread().getName()+" Ended");
    }
}

public static void main(String[] args) {
    ThreadDemo obj1 = new ThreadDemo();
    ThreadDemo obj2 = new ThreadDemo();
    obj1.setPriority(Thread.MIN_PRIORITY);
    obj1.setName("Thread1");
    obj2.setPriority(Thread.MAX_PRIORITY);
    obj2.setName("Thread2");

    obj1.start();

    try{
        obj1.join(2000);
    }
    catch (InterruptedException e) {
        System.out.println("Exception has occurred" + e);
    }

    obj2.start();
}
}
```



```
<terminated> ThreadDemo [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 5:05:22 AM – 5:05:30 AM) [pid: 29636]
Thread1 is running...
Priority is : 1
Name is : Thread1
User Thread1
User Thread1
Thread2 is running...
Priority is : 10
Name is : Thread2
User Thread2
User Thread1
User Thread2
User Thread1
User Thread2
User Thread1
User Thread2
Thread1 Ended
User Thread2
Thread2 Ended
```

Viva Questions :

****Question 1:**** What is multithreading in Java, and why is it used?

****Answer:**** Multithreading in Java is a mechanism that allows a program to execute multiple threads concurrently, enabling it to perform multiple tasks simultaneously. It is used to improve program performance, responsiveness, and efficiency in handling tasks that can be parallelized.

****Question 2:**** How can you create a thread in Java?

****Answer:**** You can create a thread in Java by:

1. Extending the `Thread` class and overriding its `run()` method.
2. Implementing the `Runnable` interface and passing an instance of the implementing class to a `Thread` object's constructor.

****Question 3:**** What are the advantages of implementing multithreading in a Java program?

****Answer:**** The advantages of implementing multithreading in Java include:

- Improved program responsiveness.
- Enhanced performance on multi-core processors.
- Efficient utilization of system resources.
- Better support for concurrent and parallel tasks.

****Question 5:**** What is the `start()` method in Java threads, and how does it differ from the `run()` method?

****Answer:**** The `start()` method is used to begin the execution of a thread, and it invokes the `run()` method of the thread. The `run()` method contains the code that runs in the thread's context. Calling `run()` directly does not start a new thread; it runs in the current thread's context.

****Question 7:**** What is the purpose of the `sleep()` method in Java threads?

****Answer:**** The `sleep()` method is used to pause the execution of a thread for a specified amount of time (in milliseconds). It is often used to introduce delays in thread execution or to implement time-based behavior.

****Question 8:**** What does the `join()` method do in Java threads?

****Answer:**** The `join()` method is used to wait for a thread to finish its execution before proceeding with the rest of the program. It allows one thread to "join" the execution of another thread, ensuring that they synchronize their execution.

****Question 9:**** How can you interrupt the execution of a Java thread?

****Answer:**** You can interrupt the execution of a Java thread by calling the `interrupt()` method on the thread object. This sets the thread's interrupt status flag, which can be checked within the thread's code to gracefully handle interruption.

****Question 10:**** What is thread priority in Java, and how is it set?

****Answer:**** Thread priority in Java is used to indicate the relative importance of one thread compared to others. Thread priorities are integers ranging from 1 (lowest) to 10 (highest). You can set a thread's priority using the `setPriority()` method, and you can query its priority using the `getPriority()` method. However, thread priorities are platform-dependent, and their use should be limited due to platform variability.

8. Write a JAVA program to create and import packages in JAVA

```
package shapes;

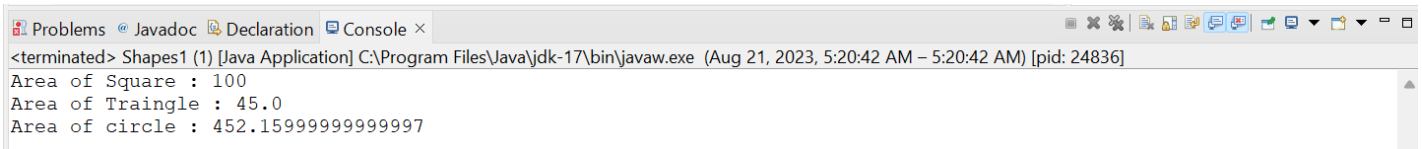
class square {
    int a, A;
    public void inputvalues(int x) {
        a=x;
    }
    public void displayArea() {
        A = a*a;
        System.out.println("Area of Square : "+A);
    }
}

class triangle {
    double a, b, A;
    public void inputvalues(int x , int y) {
        a=x;
        b=y;
    }
    public void displayArea() {
        A = 0.5*a*b;
        System.out.println("Area of Traingle : "+A);
    }
}

class circle {
    double r, A;
    public void inputvalues(int x ) {
        r=x;
    }
    public void displayArea() {
        A = 3.14*r*r;
        System.out.println("Area of circle : "+A);
    }
}

package shapes;

public class Shapes1 {
    public static void main(String[] args) {
        square s = new square();
        triangle t = new triangle();
        circle c = new circle();
        s.inputvalues(10);
        s.displayArea();
        t.inputvalues(9,10);
        t.displayArea();
        c.inputvalues(12);
        c.displayArea();
    }
}
```



```
<terminated> Shapes1 (1) [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Aug 21, 2023, 5:20:42 AM - 5:20:42 AM) [pid: 24836]
Area of Square : 100
Area of Traingle : 45.0
Area of circle : 452.15999999999997
```

Viva Questions :

****Question 1:**** What is a Java package, and why is it used?

****Answer:**** A Java package is a way to organize related classes and interfaces into a single namespace. It is used to group classes that belong together and prevent naming conflicts with classes in other packages.

****Question 2:**** How do you declare a package in a Java source file?

****Answer:**** To declare a package in a Java source file, you use the ``package`` statement at the beginning of the file.
For example:

```
package com.mycompany.myapp;
```

****Question 3:**** What is the default package in Java?

****Answer:**** The default package in Java is the package that does not have a specified name. Classes without a ``package`` statement are part of the default package. It's generally recommended to avoid using the default package for larger projects.

****Question 4:**** What is the purpose of creating custom packages in Java?

****Answer:**** The purpose of creating custom packages in Java is to organize classes into a logical and structured hierarchy, making it easier to manage and maintain code. Custom packages also help prevent naming conflicts and improve code reusability.

****Question 5:**** How do you import classes from a package in Java?

****Answer:**** You can import classes from a package in Java using the ``import`` statement. For example:

```
import com.mycompany.myapp.MyClass;
```

****Question 6:**** What is the difference between importing a specific class and importing an entire package in Java?

****Answer:**** Importing a specific class allows you to use that class without qualifying its name with the package name. Importing an entire package makes all the classes and interfaces within that package available for use without qualification.

****Question 7:**** How do you access a class in a different package in Java?

****Answer:**** To access a class in a different package in Java, you can:

- Use the fully qualified class name (e.g., ``com.mycompany.myapp.MyClass``) without importing it.
- Import the class using the ``import`` statement and then use its simple name (e.g., ``import com.mycompany.myapp.MyClass;`` and then use ``MyClass``).