Name: Abhishek Srivastava.
Reg. no.: 19BCE10071

Subject: Programming in Java
Date: Jan 25, 2021
Faculty: Dr. Shriram R
Class No.: 1034 / B11

Answers

(1)

(b)

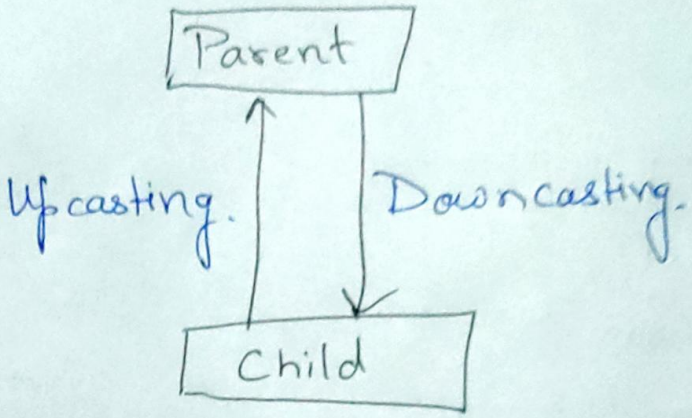
Typecasting is a concept that deals with the conversion of one type of data into another type implicitly or explicitly.

Upcasting.

It is the typecasting of a child object to a parent object. Upcasting can be done implicitly. It gives us flexibility to access the parent class members but its not possible to access all child class members using this feature.

Downcasting.

Similar to upcasting, downcasting means the typecasting of a parent object to a child object.
Downcasting cannot be implicit.



Example Program.

```
class Parent {
    String name;
    void method () {
        System.out.println ("From Parent");
    }
}

class child extends Parent {
    int id;
    void method () {
        System.out.println ("From Child");
    }
}

public class Monitor {
    public static void main (String args[]) {
        Parent p = new Child(); // Upcasting.
        p.name = "Random";

        Child c = (Child) p; // Downcasting.
        c.id = 1;
    }
}
```

(3) (a)

Exceptional Handling.

Some of the approaches to handle exception in java are :-

- try... catch block.
- finally block
- throw and throws Keyword.

a] try... catch block

Syntax :-

```
try {
    // code
}
catch (Exception e) {
    //code
}
```

Here, when an exception occurs, it is caught by the catch block. The catch block cannot be used without a try block.

Example

class

P T O

```
class Main {
    public static void main (String args[]) {
        try {
            int divideByZero = 5/0;
        }
        catch (ArithmeticException e){
            System.out.println ("Error: " + e.getMessage());
        }
    }
}
```

Output :- Error: / by zero.

Here, we have tried dividing by 0, so this throws an error which is printed in catch block.

b) Java finally block.

Syntax:-

```
try {
    // code
}
catch (ExceptionType e1) {
    // code catch block
}
finally {
    // finally block always executes
}
```

The finally block is always executed no matter whether there is an exception or not. This block is optional, and for each try block, there can only be one finally block.

If an error occurs, the finally block is executed after try... catch block, otherwise, it is executed after try block.

Example

```
class Main {
    public static void main (String args[]) {
        try {
        int divideByZero = 5/0;
        }
        catch (ArithmeticException e) {
            System.out.println("Error : " + e.getMessage());
        }
        finally {
            System.out.println("This is finally block");
        }
    }
}
```

Output :- Error : / by zero
This is finally block.

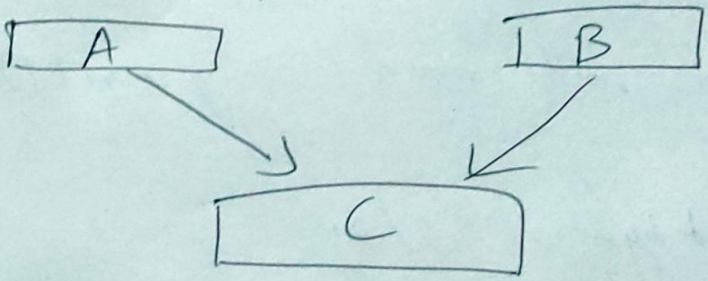
4]

Multiple Inheritance.

Multiple Inheritance is a feature of object oriented concept where a class can inherit properties of more than one parent class.

Java doesn't support Multiple Inheritance.

To implement the functionality of ~~multi~~ Multiple Inheritance in java, we use interfaces.




```
public class Vehicle {
    public string name = "";
    public string color = "";
    public string company = "";
    public string engine = "";
    public Vehicle ( String n, String c, String cmp )
    {
        name = n;
        color = c;
        company = cmp;
    }
```

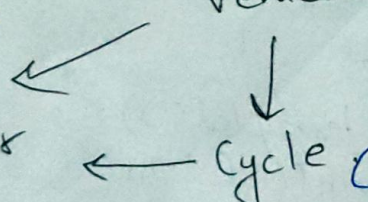
~~public class Main {~~

public

```
public Vehicle (String n, String c, String cmp, String enj)
{
    name = n;
    color = c;
    company = cmp;
    engine = enj;
}
interface Cycle extends Vehicle {
    Cycle Cycle c = new Cycle ("A", "red", "bikey");
}
class Scooter extends Vehide implements cycle {
    Scooter s = new Scooter ("B", "Blue", "York", "500cc");
}
```

Explanation

In above Example, Cycle extends properties of Vehicle and Scooter extends properties of both Cycle and Vehicle.



5)

Stopping a Thread in Java.

A thread is automatically destroyed when the run() method has completed. But sometimes it is required to stop/kill the thread before it has completed its life cycle.

Some deprecated method to manage execution of Thread were suspend(), resume() and stop(), but they could result in System failure.

Modern ways to stop threads is using a boolean flag and Thread.interrupt() method.

Example to stop a thread using Thread.interrupt()

```
class MyThread implements Runnable {
    Thread t;
    MyThread() {
        t = new Thread(this);
        t.start();
    }
    public void run() {
        while (!Thread.interrupted()) {
            System.out.println("Thread is Running");
        }
        System.out.println("Stopped");
    }
}
```

```
public class Main {
    public static void main (String args[]) {
        MyThread t1 = new MyThread();
        try {
            Thread.sleep (1000);
            t1.t.interrupt ();
            Thread.sleep (5);
        }
        Catch (InterruptedException e) {
            System.out.println ("Caught: " + e);
        }
        System.out.println ("Existing main thread");
    }
}
```

Whenever an interrupt has been sent, the thread will stop, no matter, what task it is performing.

Difference between calling a Method run from main class and calling a thread via start.

As we know, start() and run() are two important method of multithreading and one is used to create a new Thread and other is used to start executing that thread.

Here is an Example of both to print a series of odd numbers.

Side-1

```
public class JavaTester extends Thread {
    public void run() { int n=1
        while (true) {
            System.out.print(n + ", ");
            n += 2;
        }
    }
    public static void main (String args[]){
        JavaTester t1 = new JavaTester ();
        // This will call run method.
        t1.start();
    }
}
```

Side-2

```
public class Javatester implements Runnable {
    public void run () {
        int n=1
        while (true) {
            System.out.print (n + ", ");
            n += 2;
        }
    }
    public static void main (String args[]){
        Javatester m1 = new Javatester ();
        Thread t1 = new Thread (m1);
        // This will call run method.
        t1.start ();
    }
}
```

2) (b)

Web streams to get content of a Website:-

openStream() method.

The URL class of java.net package represents a Uniform Resource Locator which is used to point a resource in world wide web.

The openStream() method of this class opens a connection to the URL represented by the current object and returns an InputStream object using which you can read data from the URL.

Program

```
import java.net.URL;
import java.util.Scanner;
public class ReadingWeb {
    public static void main (String args[]) {
        URL url = new URL("https://www.java.com/");
        // Retrieving contents of specified page.
        Scanner sc = new Scanner(url.openStream());
        // Initializing the StringBuffer class.
        StringBuffer sb = new StringBuffer();
```

```
while (sc.hasNext()){
    sb.append(sc.next());
}
// Retriving the string from String Buffer object
String result = sb.toString();
System.out.println(result);
}
}
```

Use cases of program

There are numerous use cases of reading content from web page and open Stream. Some of them are listed below:-

- Gathering data from multiple resources for analysis
- For marketing: Lead Generation.
- Price Comparism / Competative Monitoring.
- Collecting training and testing data for ML.