



NAME: Fareeha Jehangiri

ID#: 16051

MODULE: Bachelors {Software Engineering}

SEMESTER: 2nd

SECTION: A

SUBJECT: Object Oriented Programming

INSTRUCTOR: M. Ayub Khan

DATE: 29/JUNE/2020

Q1. a. Why access modifiers are used in java, explain in detail Private and Default access modifiers?

ANSWER:

Java Access Modifiers – Public, Private, Protected & Default:

You must have seen public, private and protected keywords while practising java programs, these are called access modifiers. An access modifier restricts the access of a class, constructor, data member and method in another class. In java we have four access modifiers:

1. default
2. private
3. protected
4. Public

Private access modifiers:

Private: The private access modifier is specified using the keyword **private**. The methods or data members declared as private are accessible only **within the class** in which they are declared. Any other **class of same package will not be able to access** these members. Top level Classes or interface can not be declared as private because private means “only visible within the enclosing class”.

protected means “only visible within the enclosing class and any subclasses”

Hence these modifiers in terms of application to classes, they apply only to nested classes and not on top level classes.

Default access modifiers:

Default: When no access modifier is specified for a class, method or data member – It is said to be having the **default** access modifier by default.

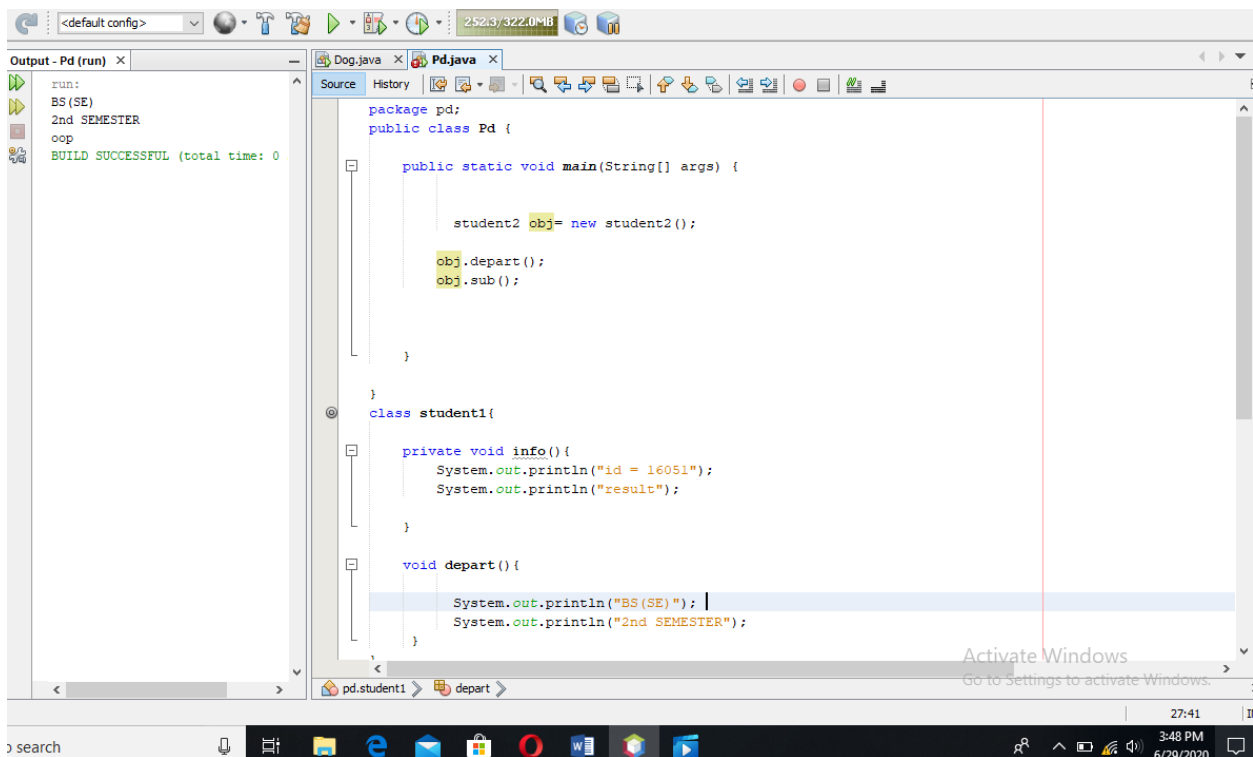
The data members, class or methods which are not declared using any access modifiers i.e. having default access modifier are accessible **only within the same package**.

When we do not mention any access modifier, it is called default access modifier. The scope of this modifier is limited to the package only. This means that if we have a class with the default access modifier in a package, only those classes that are in this package can access this class. No other class outside this package can access this class. Similarly, if we have a default method or data member in a class, it would not be visible in the class of another package. Let's see an example to understand this.

b. Write a specific program of the above-mentioned access modifiers in java.

ANSWER:

Program in NetBeans with output:



The screenshot shows the NetBeans IDE interface. On the left, the 'Output' window displays the following text:

```
run:
BS (SE)
2nd SEMESTER
oop
BUILD SUCCESSFUL (total time: 0
```

The main editor window shows the source code for a Java program:

```
package pd;
public class Pd {

    public static void main(String[] args) {

        student2 obj = new student2();

        obj.depart();
        obj.sub();

    }

}

class student1{

    private void info(){
        System.out.println("id = 16051");
        System.out.println("result");
    }

    void depart(){

        System.out.println("BS (SE) ");
        System.out.println("2nd SEMESTER");
    }

}
```

The bottom status bar shows the current file is 'pd.student1' and the cursor is at the 'depart' method. The system tray at the bottom right indicates the time is 3:48 PM on 6/29/2020.

```
run:
BS (SE)
2nd SEMESTER
oop
BUILD SUCCESSFUL (total time: 0)

void depart() {
    System.out.println("BS (SE)");
    System.out.println("2nd SEMESTER");
}

class student2 extends student1 {
    void sub() {
        System.out.println("oop");
    }

    void info() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated me
    }
}
```

Q2. a. Explain in detail Public and Protected access modifiers?

ANSWER:

Public Access Modifier:

A class, method, constructor, interface, etc. declared public can be accessed from any other class. Therefore, fields, methods, blocks declared inside a public class can be accessed from any class belonging to the Java Universe.

However, if the public class we are trying to access is in a different package, then the public class still needs to be imported. Because of class inheritance, all public methods and variables of a class are inherited by its subclasses.

Protected Access Modifier:

Variables, methods, and constructors, which are declared protected in a superclass can be accessed only by the subclasses in other package or any class within the package of the protected members' class.

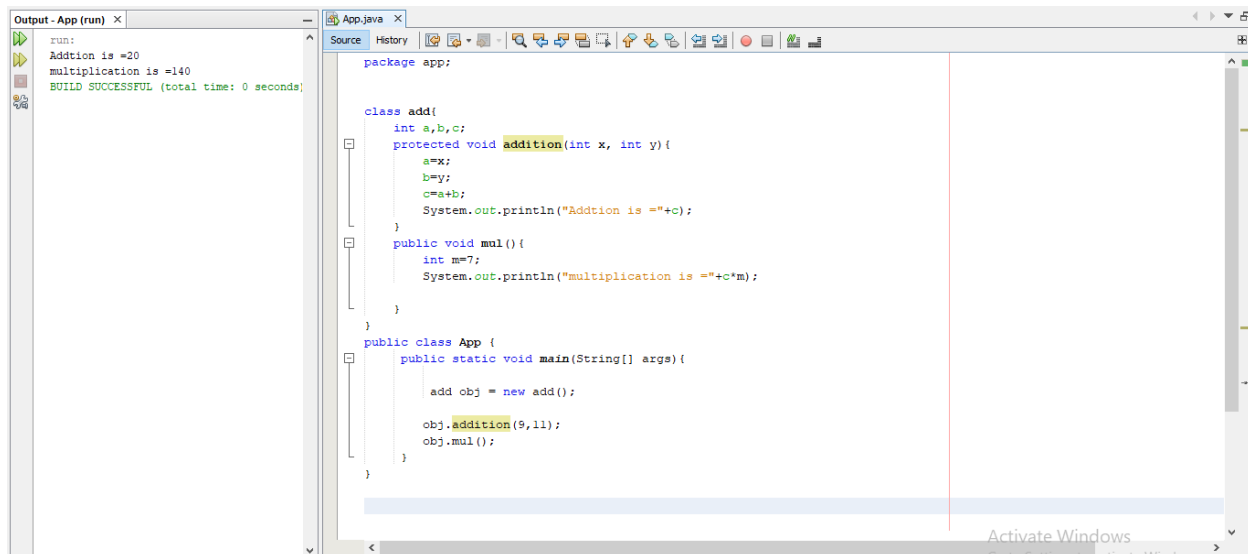
The protected access modifier cannot be applied to class and interfaces. Methods, fields can be declared protected, however methods and fields in a interface cannot be declared protected.

Protected access gives the subclass a chance to use the helper method or variable, while preventing a nonrelated class from trying to use it.

b. Write a specific program of the above-mentioned access modifiers in java.

ANSWER:

Program in NetBeans with output:



The screenshot shows the NetBeans IDE with a Java program in the 'App.java' file. The program defines a class 'add' with a protected method 'addition' and a public method 'mul'. The 'App' class uses these methods. The output window shows the results of running the program.

```
package app;

class add{
    int a,b,c;
    protected void addition(int x, int y){
        a=x;
        b=y;
        c=a+b;
        System.out.println("Addition is "+c);
    }
    public void mul(){
        int m=7;
        System.out.println("multiplication is "+c*m);
    }
}

public class App {
    public static void main(String[] args){
        add obj = new add();

        obj.addition(9,11);
        obj.mul();
    }
}
```

Output - App (run) x

```
run:
Addition is =20
multiplication is =140
BUILD SUCCESSFUL (total time: 0 seconds)
```

Q3. a. What is inheritance and why it is used, discuss in detail?

ANSWER:

Inheritance:

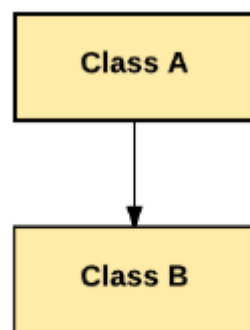
Inheritance is a mechanism in which one class acquires the property of another class. For example, a child inherits the traits of his/her parents. With inheritance, we can reuse the fields and methods of the existing class. Hence, inheritance facilitates Reusability and is an important concept of OOPs.

Types of Inheritance:

There are Various types of inheritance in Java:

Single Inheritance:

In Single Inheritance one class extends another class (one class only).

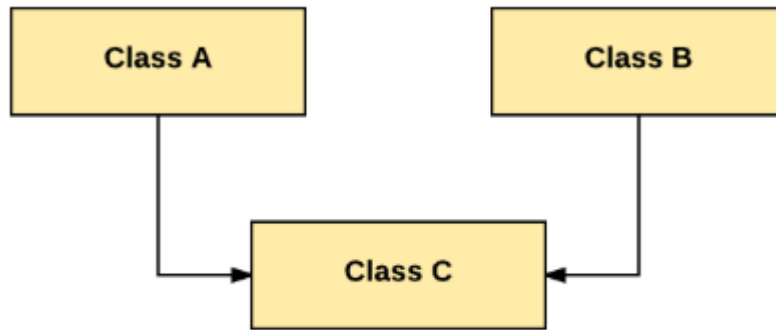


Single Inheritance

In above diagram, Class B extends only Class A. Class A is a super class and Class B is a Sub-class.

Multiple Inheritance:

In Multiple Inheritance, one class extending more than one class. Java does not support multiple inheritance.

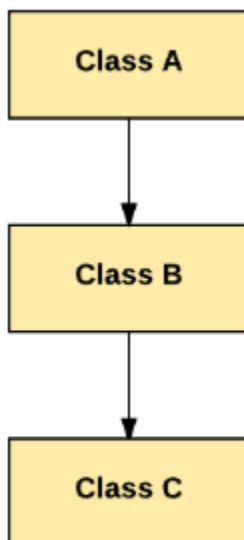


Multiple Inheritance

As per above diagram, Class C extends Class A and Class B both.

Multilevel Inheritance:

In Multilevel Inheritance, one class can inherit from a derived class. Hence, the derived class becomes the base class for the new class.

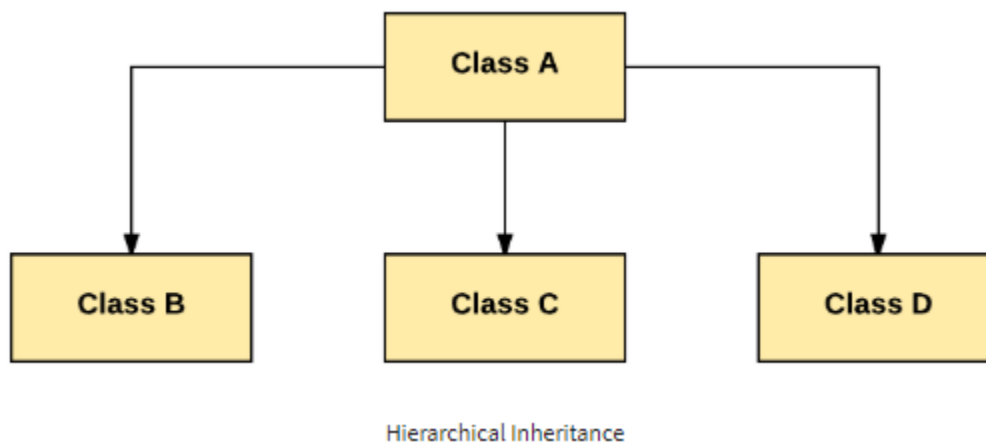


Multilevel Inheritance

As per shown in diagram Class C is subclass of B and B is a of subclass Class A.

Hierarchical Inheritance:

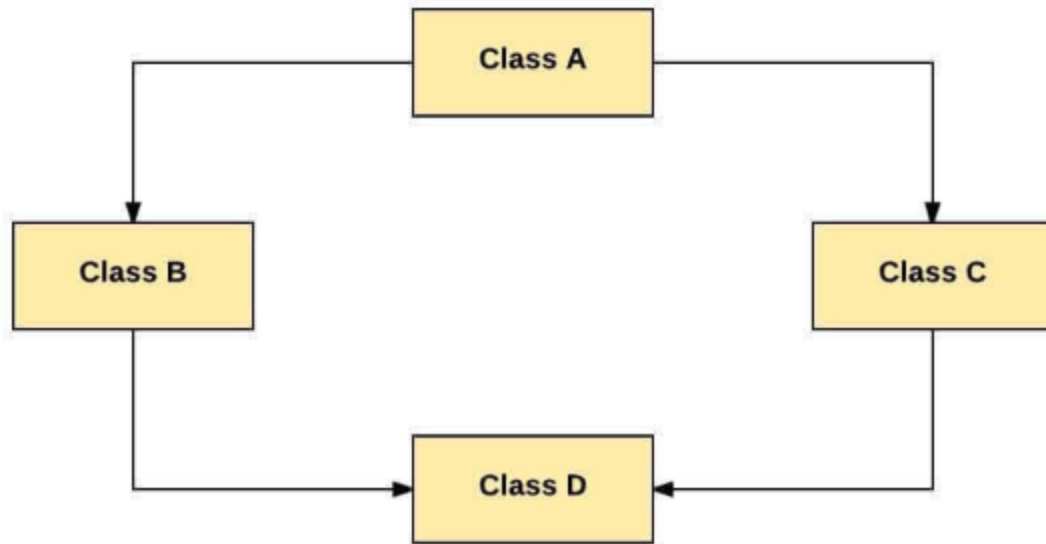
In Hierarchical Inheritance, one class is inherited by many sub classes



As per above example, Class B, C, and D inherit the same class A.

Hybrid Inheritance:

Hybrid inheritance is a combination of Single and Multiple inheritance.



As per above example, all the public and protected members of Class A are inherited into Class D, first via Class B and secondly via Class C.

Note: Java doesn't support hybrid/Multiple inheritance

Inheritance in Java

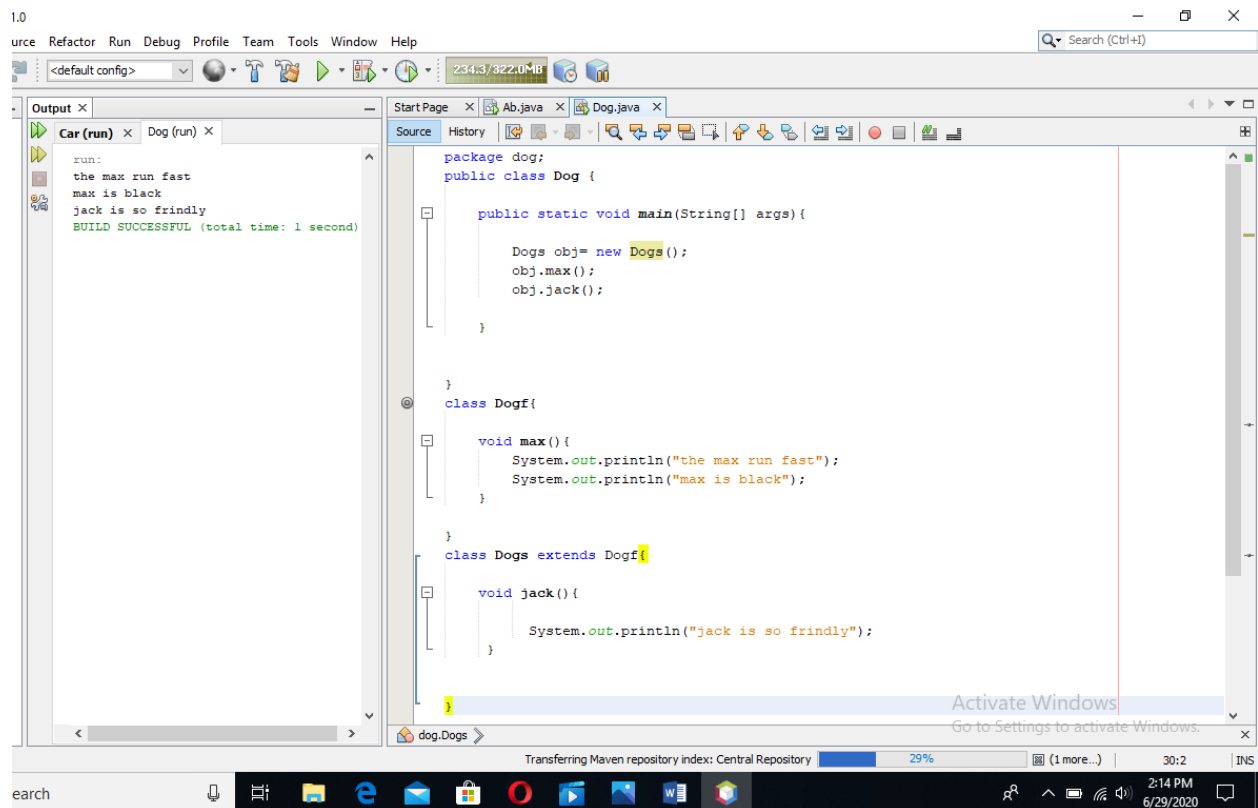
JAVA INHERITANCE: is a mechanism in which one class acquires the property of another class. In Java, when an "Is-A" relationship exists between two classes, we use Inheritance. The parent class is called a super class and the inherited class is called a subclass. The keyword `extends` is used by the sub class to inherit the features of super class.

Inheritance is important since it leads to the reusability of code

b. Write a program using Inheritance class on Animal in java.

ANSWER:

Program in NetBeans with output:



Q4. a. What is polymorphism and why it is used, discuss in detail?

ANSWER:

Polymorphism:

Polymorphism has 2 requirements:

- A subclass method must override super-class method. Only super-class method with public and protected scope are eligible for overriding.

- A super class reference is used to invoke the method.

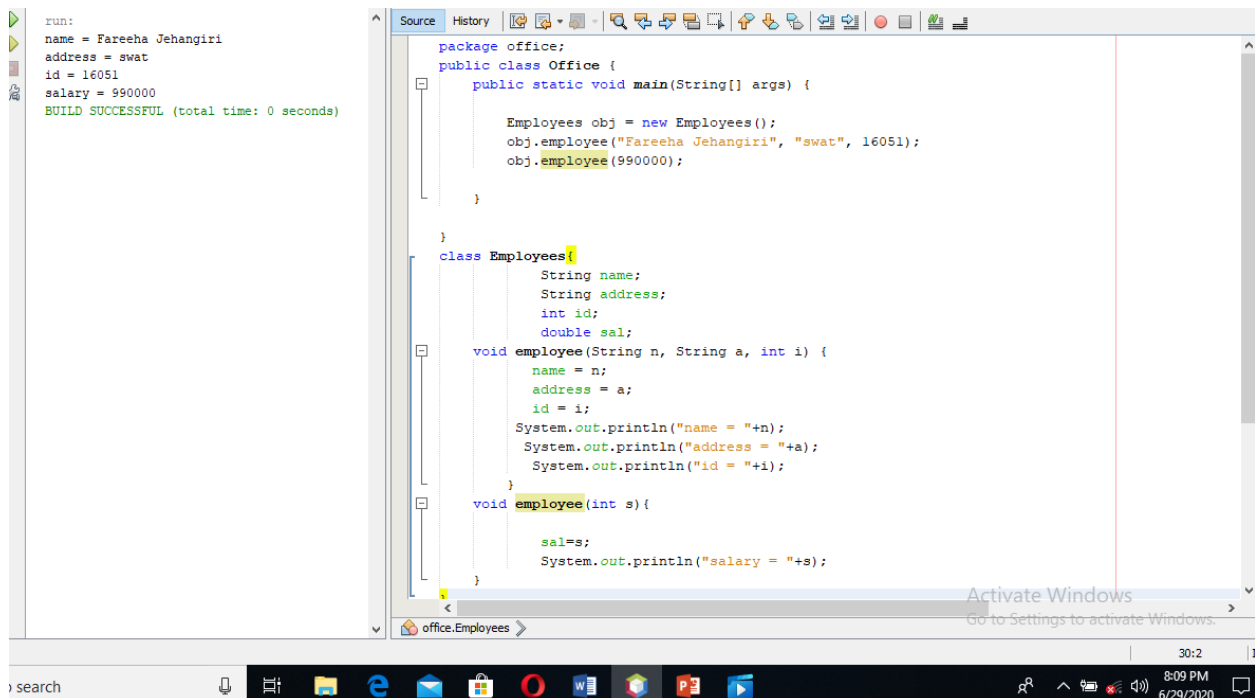
Example

- **class** Employee **extends** Office
- {
- Office obj = new Employee();
- obj.print();
- }
- Here **obj** polymorphically invokes the **print()** method.

b. Write a program using polymorphism in a class on Employee in java.

ANSWER:

Program in NetBeans with output:



```
run:
name = Fareeha Jehangiri
address = swat
id = 16051
salary = 990000
BUILD SUCCESSFUL (total time: 0 seconds)

package office;
public class Office {
    public static void main(String[] args) {
        Employees obj = new Employees();
        obj.employee("Fareeha Jehangiri", "swat", 16051);
        obj.employee(990000);
    }
}

class Employees{
    String name;
    String address;
    int id;
    double sal;
    void employee(String n, String a, int i) {
        name = n;
        address = a;
        id = i;
        System.out.println("name = "+n);
        System.out.println("address = "+a);
        System.out.println("id = "+i);
    }
    void employee(int s){
        sal=s;
        System.out.println("salary = "+s);
    }
}
```

Q5. a. Why abstraction is used in OOP, discuss in detail?

ANSWER:

Abstraction:

Abstraction is selecting data from a larger pool to show only the relevant details of the object to the user. Abstraction “shows” only the essential attributes and “hides” unnecessary information. It helps to reduce programming complexity and effort. It is one of the most important concepts of OOPs.

In simplest words, you can define abstraction as which captures only those details about a Java object that are relevant to the current perspective.

For example, a `HashMap` stores key-value pairs. It provides you two methods `get()` and `put()` methods to store and retrieve key-value pairs from map. It is, in fact, the only information you will need if you want to use the map in your application. How it works inside, you are not required to know it to use it. This is very much **example of abstraction in Java**.

b. Write a program on abstraction in java.

ANSWER:

Program in NetBeans with output:

The screenshot displays an IDE window with the following components:

- Output - subtraction (run):** Shows the execution result: `run: subtraction is=7 BUILD SUCCESSFUL (total time: 0 s)`
- Source:** Contains the Java code for `Subtriction.java`:

```
package subtraction;
public class Subtriction {

    public static void main(String[] args) {
        B obj = new B();
        obj.def(10,3);
        obj.sub();
    }

    abstract class A{
        abstract void sub();

        int a,b,c;
        void def(int x, int y){
            a=x;
            b=y;
        }
    }

    class B extends A{

        void sub(){
            c=a-b;
            System.out.println("subtriction is="+c);
        }
    }
}
```
- Taskbar:** Shows the Windows taskbar with the search bar, Start menu, and various application icons. The system tray on the right indicates the time is 5:26 PM on 6/29/2020.