CS261

ASSIGNMENT 4

NAME:

ARCHIT AGRAWAL

ROLL NO. :

202052307

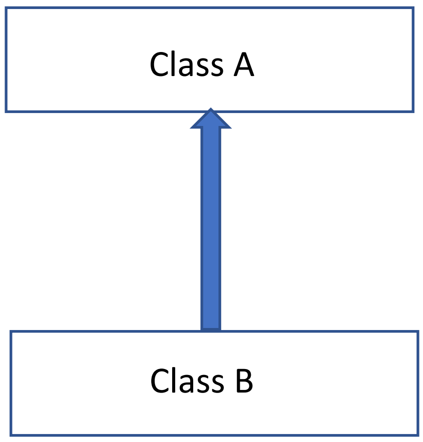
SECTION:

A

1. **Draw a diagram of single inheritance. Write a program to find areas and volume of BEDROOM in which class “Bedroom” is extended from class “Room”.**

In single inheritance, the derived class inherits the single base class either publicly, privately or protectedly. The derived (or child) class uses the features or members of single base (or parent) class. These base class members or attributes can be accessed by the derived class according to the access specifier specified during inheriting the base class.

A single inheritance diagram looks like this.

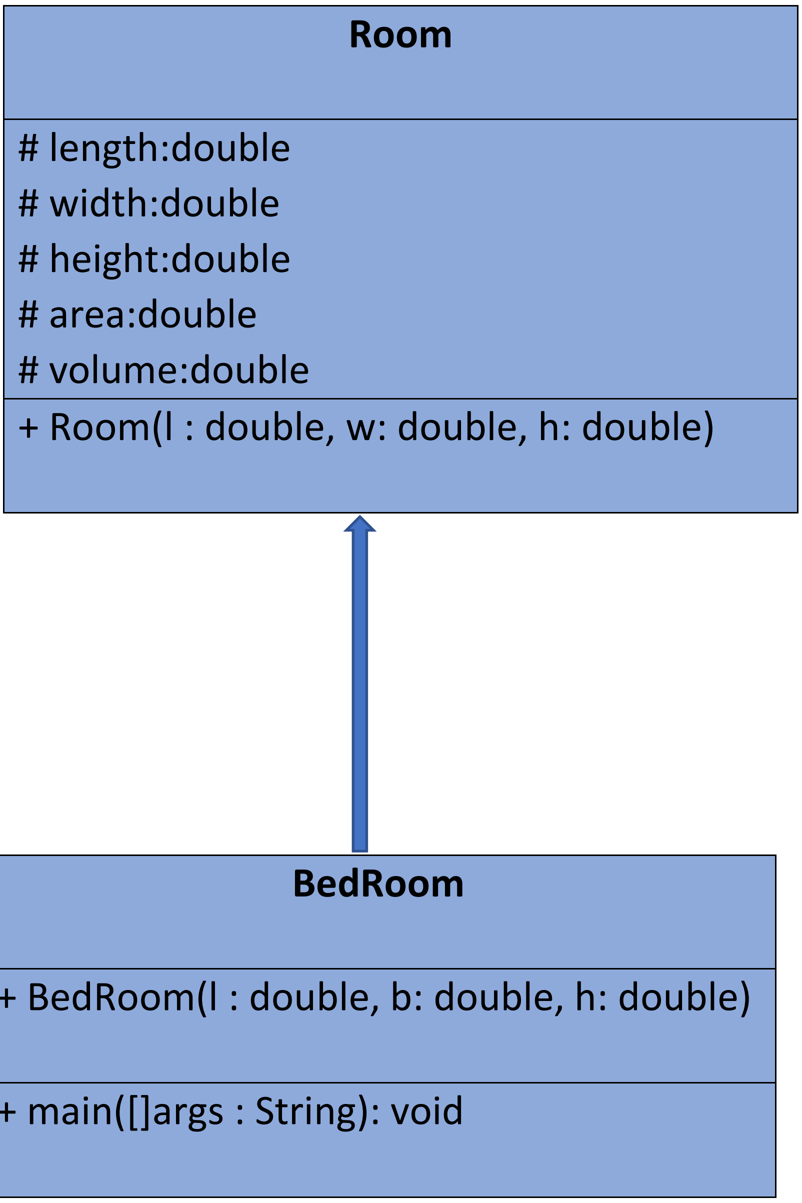


Class B (Child) is inherited from Class A (parent).

<u>Approach for writing the Program</u>

Here, we created a class Room. Since, every room has a certain length, breadth, height, area and volume. All these properties are instanced in class Room itself. Next, we made a constructor to set the length, breadth and height of room. Once the length, breadth and height are provided, the area and volume are fixed and they can’t be changed. The area and volume are instantiated using provided length, breadth and height values.

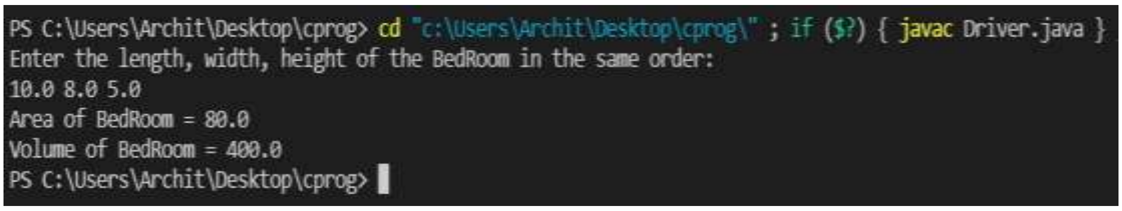
Next, we created a class BedRoom which inherits the class Room. Hence, this class has all the attributes and members of the class Room. Now, we want to instantiate the instance variables of object of BedRoom class. To do so, we can take length, breadth and height as input and we can call the constructor of base class i.e. the class Room using ***super*** keyword. This way we don't need to write the constructor again for BedRoom class.


Room
# length:double
# width:double
# height:double
# area:double
# volume:double
+ Room(l : double, w: double, h: double)
BedRoom
+ BedRoom(l : double, b: double, h: double)
+ main([]args : String): void

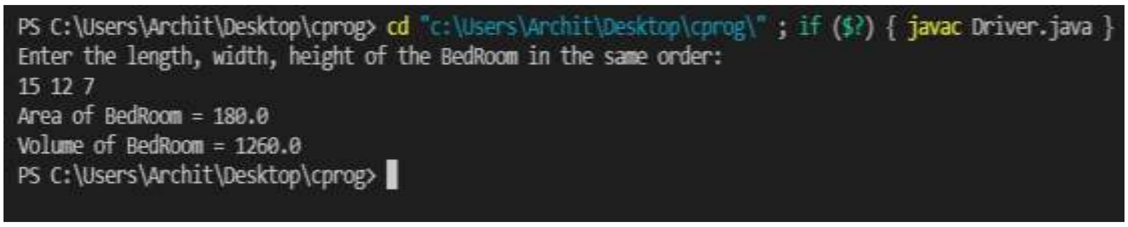
CODE

```java
import java.util.*;

//class Room
class Room {

    //instance variables
    protected double length;
    protected double width;
    protected double height;
    protected double area;
    protected double volume;

    //instance variable are declared protected
    //so that the derived class can access these instance variables

    //constructor
    //access specifier is public because object of Room Class can be
made from anywhere
    public Room(double l, double w, double h){
        this.length = l;
        this.width = w;
        this.height = h;
        this.area = l * w;
        this.volume = l * w * h;
    }
}

public class BedRoom extends Room{

    //constructor
    //access specifier is public because object of the class can be
made from anywhere
    public BedRoom(double l, double b, double h){
        super(l, b, h);
    }

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        //creating a new object of BedRoom class

        //Taking length, width and height input from user
        System.out.println("Enter the length, width, height of the
BedRoom in the same order: ");
        double l = sc.nextDouble();
        double w = sc.nextDouble();
        double h = sc.nextDouble();
        BedRoom bd = new BedRoom(l, w, h);

        System.out.println("Area of BedRoom = "+bd.area);
        System.out.println("Volume of BedRoom = "+bd.volume);
    }

}
```

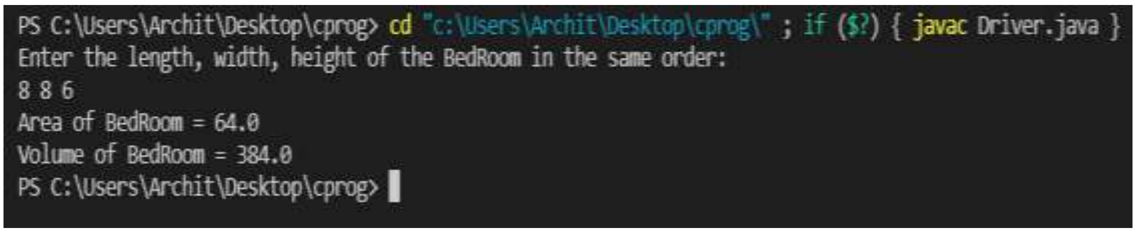
OUTPUT



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Driver.java }
Enter the length, width, height of the BedRoom in the same order:
10.0 8.0 5.0
Area of BedRoom = 80.0
Volume of BedRoom = 400.0
PS C:\Users\Archit\Desktop\cprog> 
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Driver.java }
Enter the length, width, height of the BedRoom in the same order:
15 12 7
Area of BedRoom = 180.0
Volume of BedRoom = 1260.0
PS C:\Users\Archit\Desktop\cprog> 
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?) { javac Driver.java }
Enter the length, width, height of the BedRoom in the same order:
8 8 6
Area of BedRoom = 64.0
Volume of BedRoom = 384.0
PS C:\Users\Archit\Desktop\cprog> 
```

2. **(a) Which of the following operator cannot be overloaded? Give short explanation.**
   a) +
   b) ?:
   c) –
   d) %

The ternary operator (?:) cannot be overloaded. The ternary operator works like given below:

condition ? res1 : res2

If condition is true, res1 is evaluated else res2 is evaluated. Now, if we want to overload the ternary operator the syntax will be like this:
RT className :: Operator ?: (bool condition, RT res1, RT res2){}

Now, to overload ternary operator, both res1 and res2 will be evaluated. This is not the case with ternary operator, it just evaluates only one result, either res1 or res2.

(b) Which of the following operator can be overloaded? Give short explanation.
a) ?:
b) ::
c) .
d) ==

The assignment operator (==) can be overloaded.
The operators scope(::), member selector(.) cannot be overloaded as overloading them will make serious programming issues. For an example the sizeof operator returns the size of the object or datatype as an operand. This is evaluated by the compiler. It cannot be evaluated during runtime. So we cannot overload it. The ternary operator cannot be overloaded as described in Q. 2(a).

(c) Which of the following is/are true about constructors in JAVA?

a) Constructor name should be same as class name.
b) If you don't define a constructor for a class, a default parameter less constructor is automatically created by the compiler.
c) The default constructor calls super() and initializes all instance variables to default value like 0, null.
d) If we want to parent class constructor, it must be called in first line of constructor.

All the options are correct.

3. **What will be printed on the console and why?**

```
var point1 = new Point(1, 2);

var point2 = new Point(1, 2);

System.out.println(point1.equals(point2));
```

The output on console will be ***'false'***. This is because by default the 'equals' method compares the references of the two objects to be compared. If the reference is same, it returns 'true' else it returns 'false'. Since, we have created two different objects, hence their references are not the same. Hence, the output will be 'false'.

4. **WAP in JAVA for a string s, find the longest substring without repeating characters.**

<u>Example 1:</u>

Input: s = "abcabcbb"

Output: 3

Explanation: The answer is "abc", with the length of 3.

<u>Example 2:</u>

Input: s = "bbbbb"

Output: 1

Explanation: The answer is "b", with the length of 1.

<u>Approach to solve this problem:</u>

- If an empty string is passed, return 0.
- Convert the string to lower case to prevent any case related issues.
- Declare a String variable str, this will be used to store a string, and its maximum length will be our answer. Declare another variable max which stores maximum length of str and initialize it with -1.

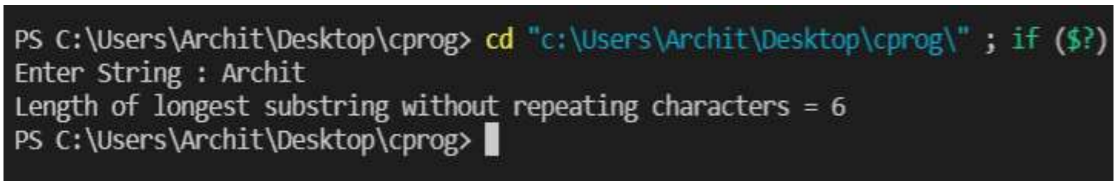
- Run a loop from i = 0 to i < s.length() and store the character at ith index in string s in a char variable ch.
- Now, if ‘ch’ is present in the string str, update str to (str + ch), and remove the first character from str. Else, update str with str + ch (+ means concatenation).
- In each iteration, compute the maximum of max and str.length() and store it in max.
- Return max.

LongestUniqueSubstring
+ longestUniqueSubstring(s: String):int + main([]args : String): void

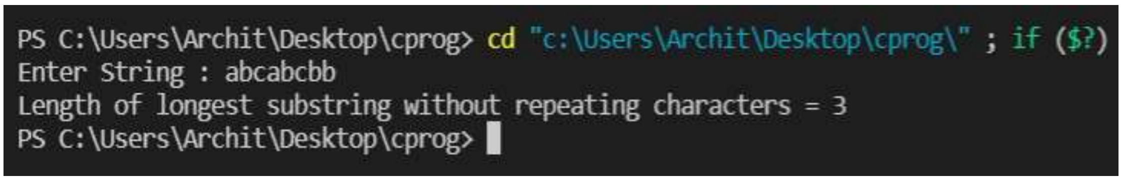
CODE

```
import java.util.*;
public class LongestUniqueSubstring{

    //method to find length of longest substring without repeating
characters
    public int longestUniqueSubstring(String s){

        //basic case, empty string is passed.
        if(s.length() == 0) return 0;
        s = s.toLowerCase();
        //str stores the longest substring
        String str = "";
        char ch;
        //initilaising maiximum length with -1
        int max = -1;

        for(int i = 0; i < s.length(); i++){
            ch = s.charAt(i);

            //if the character is not already present, we add it in str
            if(str.indexOf(ch) == -1){
                str = str + ch;
            }
            //else we update str to str + ch and remove the first 
character from str
            else{
                str = str + ch;
                str = str.substring(str.indexOf(ch) + 1);
            }

            //check and update max with maximum(max, str.length())
            max = Math.max(max, str.length());
        }

        return max;
    }

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);
        //taking input
        System.out.print("Enter String : ");
        String s = sc.nextLine();
        //creating object to call method to perform task
        LongestUniqueSubstring len = new LongestUniqueSubstring();
        //printing ouput
        System.out.println("Length of longest substring without 
repeating characters = "+len.longestUniqueSubstring(s));
    }
}
```

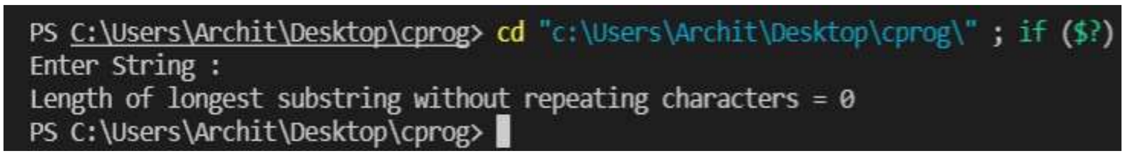
OUTPUT



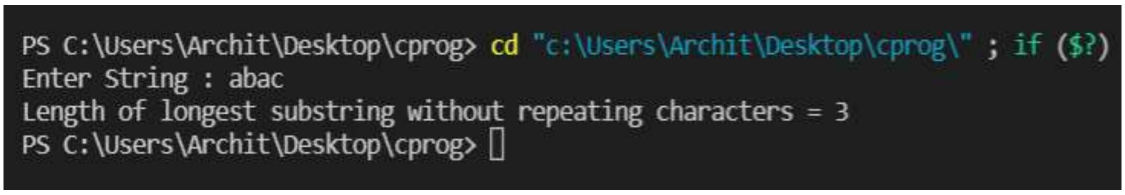
```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?)
Enter String : Archit
Length of longest substring without repeating characters = 6
PS C:\Users\Archit\Desktop\cprog> 
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?)
Enter String : abcabcbb
Length of longest substring without repeating characters = 3
PS C:\Users\Archit\Desktop\cprog> 
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?)
Enter String :
Length of longest substring without repeating characters = 0
PS C:\Users\Archit\Desktop\cprog>
```



```
PS C:\Users\Archit\Desktop\cprog> cd "c:\Users\Archit\Desktop\cprog\" ; if ($?)
Enter String : abac
Length of longest substring without repeating characters = 3
PS C:\Users\Archit\Desktop\cprog>
```