```
package javasamplepaper;
/*
(1)
(a) - Employee(),Employee(String name,float salary) = Constructors
    - raiseSalary(float percentRaise),getName(),getSalary(),displayDetails() = Methods
    - name(String),salary(float) = Data
-------------------------------------------------------------------------------------------------
*/
//(b)
class Employee {

    private String name;
    private float salary;

    public Employee()
    {
        this.name = "UnKnown";
        this.salary = 0.0f;
    }

    public Employee(String name,float salary)
    {
        this.name = name;
        this.salary = salary;
    }

    public void raiseSalary(float percentRaise)
    {
        this.salary = ((percentRaise + 100.0f) * this.salary)/100.0f;
    }

    public String getName()
    {
        return this.name;
    }

    public float getSalary()
    {
        return this.salary;
    }

    public void displayDetails()
    {
        System.out.println("Name is : " + this.name);
        System.out.println("Salary is : " + this.salary);
    }
}


public class TestEmployee {

    public static void main(String args[])
    {
        //(c)
        //i.
        Employee william = new Employee("Bill Smith",35000.00f);
        //ii.
        william.displayDetails();
        //iii.
        william.raiseSalary(12.5f);
        william.displayDetails();

    }


}
```

```java
package javasamplepaper;
/*
(2).
i.
In Java, inheritance is used when a class wants to use/inherit the features of another existing class
*/
public class A {

}
class B extends A{

}

//ii.
abstract class Student{

    public abstract void display();
}

/*
iii.
    1. default
    2. private
    3. protected
    4. public
*/

/*
iv.
    Package in Java is a mechanism to encapsulate a group of classes, sub packages and interfaces
eg:
    java.util
    java.lang
    java.io
*/

//v.
interface Paybonus{

    public void calcBonus();
}

class Employee implements Paybonus{

    @Override
    public void calcBonus() {
        //code
    }


}
```

```java
package javasamplepaper;
//(3)
//i.

/*

Overloading occurs when two or more methods in one class have the same method name
but different parameters.

Overriding means having two methods with the same method name and parameters

*/

//Overloading Example
class overloading {

    public void overload()
    {
        System.out.println("Original Body");
    }
    public void overload(int x)
    {
        System.out.println("Overloaded body with one integer parameter :" + x);
    }
    public void overload(int x,int y,String z)
    {
        System.out.println("Overloaded body with three parameters : "+x+" And "+y+" And "+z);
    }


}

//Overriding Example
class overriding{

    public void overload()
    {
        System.out.println("Original Body");
    }
}

class A extends overriding
{
    @Override
    public void overload()
    {
        System.out.println("Same method but Overrided Body");
    }
}

//ii.
/*

Java - Exceptions.
An exception (or exceptional event) is a problem that arises during the execution of a program.
When an Exception occurs the normal flow of the program is disrupted
and the program/Application terminates abnormally, which is not recommended

We can handle Exceptions using try-catch block , throws and throw Keywords

*/

//iii.
class dividedByZero{

    private int num1 = 50;
    private int num2 = 0;
    private int ans;

    public void divideWithTry()
    {
        try
        {
            ans = num1/num2;
        }
        catch(ArithmeticException ex)
        {
            System.out.println(ex);
        }
    }

}


//(4)
//(a)
public void actionPerformed (ActionEvent e)
{
    int num1,num2;
    String ans;
    num1 = Integer.parseInt(txt1.getText());
    num2 = Integer.parseInt(txt2.getText());
    ans = Integer.toString(num1+num2);
    txt3.setText(ans);
}

/*
(b)
    Multithreading in java is a process of executing multiple threads simultaneously

    Extending by Thread class or Implementing Runnable interface

(c)
    start() - Method to start the execution of a thread. Start() invokes the run() method on the Thread object.
    sleep() - That can be used to pause the execution of current thread for specified milliseconds and nanoseconds.
    setPriority() - Used to change the thread's priority.
                    Every thread has a priority which is represented by the integer number between 1 to 10.

*/
```