Suppose you have an application that has two classes: 'Printer' and 'LaserPrinter'. The LaserPrinter class extends the Printer class. However, there is an error that prevents this source code from getting compiled successfully.

1. Find & briefly explain the error
2. Explain & code two different solutions to fix the error without removing anything from both classes.

```
public class Printer {
    private String maker;
    private int pages;

    public Printer(String maker, int pages) {
        this.maker = maker;
        this.pages = pages;
    }
}
```

```
public class LaserPrinter extends Printer{
    private String cartridgeType;
    public String getCartridgeType() {
        return cartridgeType;
    }
    public void setCartridgeType(String cartridgeType) {
        this.cartridgeType = cartridgeType;
    }
}
```

The subclass LaserPrinter does not have a constructor. The constructor cannot be inherited from the super class Printer.

Solution 1: add a constructor (as following) into the class LaserPrinter by calling the superclass constructor

```
public LaserPrinter(String maker, int pages, String cartridgeType) {
    super(maker, pages);
    this.cartridgeType = cartridgeType;
}
```

```
Solution 2: add a constructor (as following):
public LaserPrinter(String maker, int pages, String cartridgeType) {
    this.maker=maker;
    this.pages=pages;
    this.cartridgeType = cartridgeType;
}
```

solution 2 incorrect

```
solution1: in LaserPrinter class:
public LaserPrinter (String maker, int pages){
    super(maker, pages);
}
```

```
Solution2: in Printer class: construct a no parameter constructor
public Printer( ){
    this.maker=null;
    this.pages=0;
}
```

Q2) (4m)

List two methods that a subclass cannot inherit from its superclass? and why?

1. Private methods cannot be inherited. This is because when the private keyword is used for methods, these methods are restricted to access in the defining class only and not visible in its subclass.

2. final methods cannot be inherited. As long as a method has been declared as final, it cannot be overridden by any subclasses. Incorrect!

Constructor (a special method) cannot be inherited.

a. It would have the wrong name
b. The inherited constructor would not contain code to initialise any new instance variables in the subclass (initialising instance variables is a constructor's main job)

Q3) (4m)

Briefly explain the difference between Overloading and Overriding in Java. Support your answer with an example.

1. Method overloading occurs in the same class but method overriding occur in two classes (a class and its subclass) where inheritance is involved.

2. Method overloading requires methods to have different data types of parameters or different number of parameters or different order of data types of parameters, but method overriding requires the same ordered list of data types of parameters.

3. A constructor can be overloaded like methods but a constructor cannot be overridden.

4. For method overloading, the return type can be different but for method overriding, return type must be the same.

5. Overloading is used when we need multiple methods with different parameters but the methods do similar tasks. Overriding allows a class to inherit from a superclass whose behaviour is very close to its behaviour but some modifications are needed.

For example,

this is method overloading:

```
public class Test {
    public static void main(String[] args) {
        System.out.println(sum(1,2));
        System.out.println(sum(1.0,2.0));
    }
    public static int sum(int a, int b){
        return a+b;
    }

    public static double sum (double a, double b){
        return a+b;
    }
}

The output is:
3
3.0
```

From the methods coded above, it can be seen that the two methods are both called sum, but one returns data type int and the other one returns data type double. Although the parameters of these two methods are called a and b respectively, their data types are different.

```
The following is method overriding:

public class Cat{
    public void sound(){
        System.out.println("meow");
    }
}

public class Lion extends Cat{
    public void sound(){
        System.out.println("roar");
    }
}

From the methods coded above, the subclass is Lion and superclass is Cat. The
methods are both called sound, and the parameters of the methods are both empty.
As they both have void keyword, they don't return any value.
```