Q1) a) ii) "odd" will always be the output.

b) iv) Compilation ~~or~~ error (Since there is no function display())

Q2)

→ 1) What is the use of constructors?

A constructor is a block of code similar to a method. It is called when an instance of the class is executed/created. It is a special type of method which is used to initialize the object.

Types of constructors:

ⓐ Default constructor (no-arguments constructor):
A constructor without any parameter is called a default constructor.

Syntax - <class_name> () { }

ⓑ Parameterized constructor:
A constructor without any parameter is called a parameterized constructor.

Syntax - <class_name> (para1, para2, ...) { }

2) Explain the differences btw Array and Vector with examples.

→

Array	Vector
① Length is fixed	① Resizable length.
② Not synchronized.	② Synchronized.
③ Relatively fast.	③ Relatively slow
④ Does not reserve any additional storage	④ Vectors reserves additional storage.

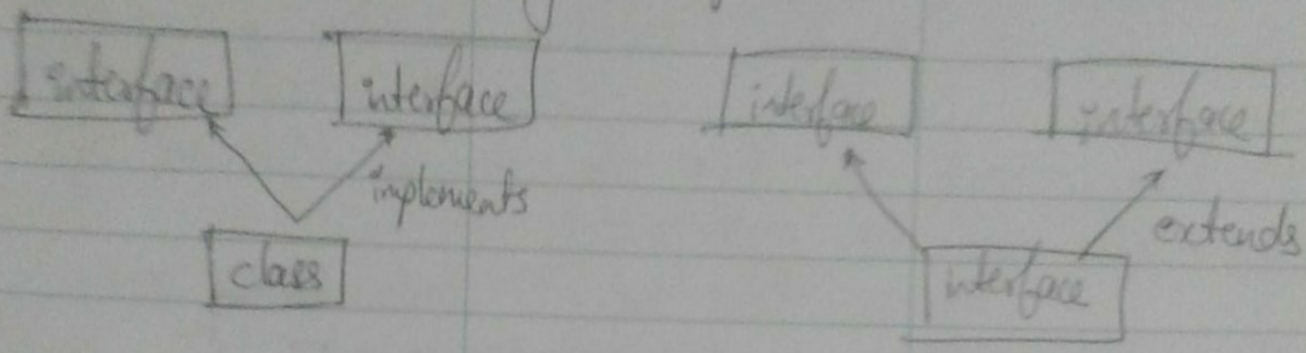
3) Difference btw Abstract class and Interface.

Abstract class	Interface
① Abstract & non abstract methods	① Only has abstract interfaces, Also default & static methods
② Does not support multiple inheritance	② Supports multiple inheritance
③ final, nonfinal static & nonstatic variables	③ only has static & final variables.
④ Provides implementation of interfaces	④ Interfaces cannot provide implementation of abstract classes

1) What is interface? Explain with examples.

→ An interface is a blueprint of a class. It has static constants and abstract methods. Interfaces help achieve abstraction. It is used to achieve abstraction and multiple ~~inter~~ inheritance in Java.

Multiple inheritance using interfaces:



If a class implements multiple interfaces or an interface extends multiple interfaces, it is known as a multiple inheritance.

Eg.
```
interface Printable{
    void print();
}
interface Showable{
    void show();
}
class A implements Printable, Showable{
public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}
public static void main(String args[]){
    A obj = new A();
    obj.print();
    obj.show();
    }
}
```

Output: Hello
Welcome.

Q5) Explain exception handling mechanism.

→ The exception handling in Java is one of the powerful mechanism to handle runtime errors so that the normal flow of the application can be maintained.

Suppose there are 10 statements in a Java programme & an exception occurs at statement 5, the rest of the code will not be executed. However, when we perform exception handling, the rest of the statements will be executed.

Eg.
```
public class JavaExceptionExample {
    public static void main (String args[]) {
        try {
            // code
            int data = 100/0;
        } catch (ArithmeticException e)
        { System.out.println (e); }
        // rest of the code.
        }
}
```

Output: Exception in thread main java.lang.ArithmeticException: / by zero.

(5)

Q6) What is a thread? Explain with examples

A thread is a lightweight subprocess, the smallest unit of processing. It is a seperate path of execution.

Threads are independent. If there occurs an exception in a thread, it doesn't affect other threads. It uses a shared memory.

Synchronization: Capability to control the access of multiple threads to any shared resource. It is a better option where we don't allow any one thread to access the shared resources.

Why use synchronization?

(a) To prevent thread interference.

(b) To prevent consistency problem.

Eg.
```
class Table{
    void printTable(int n){ // method not synchronized }
    for(int i=1; i<=5; i++){
        System.out.println(n*i);
        try{
            Thread.sleep(400);
        } catch(Exception e){ System.out.println(e);}
    }
}
}
```

```
class MyThread1 extends Thread {
    Table t;
    MyThread1 (Table t) {
        this.t = t;}
    public voic fun (){
        t.printTable (5);
    }
}
}
class MyThread2 extends Thread {
    Table t;
    MyThread2 (Table t) {
        this.t=t;}
    public void fun () {
        t.printTable (100);
    }
}
class TestSynchronization1 {
    public static void main (String args[]) {
        Table obj = new Table (); // Only one object.
        MyThread1 t1 = new MyThread1 (obj);
        MyThread2 t2 = new MyThread2 (obj);
        t1.start();
        t2.start();
    }
}
```

Output:

```
5
100
10
200
15
300
20
400
25
500
```

Output with Synchronization

```
5
10
15
20
25
100
200
300
400
500
```

Q4) @ Explain final variable with examples

→

final variable: A final variable's value cannot be changed.

Eg.
```
class Bike {
    final int speedlimit = 90;
    void fun(){
        speed limit = 400; }

    public static void main (String args[]){
        Bike obj = new Bike();
        obj.sum();
    }
}
```

Output: Compile Time error.

(b) Explain Java final method with example.

Java final Method: A final method cannot be overridden.

Eg.
```
class Bike {
    final void run() { System.out.println("running"); }
}

class Honda extends Bike {
    void run() { System.out.println("running safely"); }

    public static void main(String args[]) {
        Honda honda = new Honda();
        honda.run();
    }
}
```

Output: Compile time error.

(c) Explain Java final class with example

Java final class: A final class cannot be extended

Eg. final class Bike {}

```
class Honda extends Bike {
void run() { System.out.println("running"); }

public static void main(String args[]){
Honda honda = new Honda();
honda.run();
}
}
```

Ans: Compile time error.

* * * * * *