Test 2

1) Output

Both trainer are same

2) Exception and Exception Handling

Exception is an event that occurs by an abnormal condition in a program. For example : Dividing a number by zero can cause an exception.
When an exception occurs, the program stops abruptly and does not complete its entire execution.
In case an exception occurs, it is important to make changes in the program that notify us of the exception and complete the entire program execution. For this, we make use of Exception Handling.

Exception Handling is done by covering up the statements that may cause an exception in a try, catch block.
Eg :

```
class Add {
a=12;
b=0;
try {
        System.out.println("Result :" +(a/b));
}
catch (Exception e) {
        e.toString();
}
}
```

In the above example, dividing a by b gives an exception and it is dealt with in the catch block, where the exception is passed as an argument 'e' and is made to perform some function.

Another way to handle exceptions is by using the keywords throw and throws.
For eg:

```
class Example {
        method1() {
                System.out. println("Inside method1");
                try { method2(); }
                Catch (Exception e) { e.toString(); }
}
        method2() throws Exception {
                System.out. println("Inside method2");
                method3();
}
        method3() throws Exception {
                System.out. println("Inside method3");
                throw new Exception;
}
        public static void main(String[] args) {
                method1();
        }
}
```

In the above example the exception is thrown to be handled later. This can be done when multiple methods are used in a class.

Another way is to make use of user-defined exceptions

3) Custom Exceptions

A custom exception is a user-defined exception that can be used to handle exceptions that we know will occur in a program. For eg : A program that checks ID's of people and does not allow those below the age of 21 to enter a party.
In this case, an exception that may occur while checking ID's is that they may be of the age below 21. Our custom or user-defined exceptions provide a way to deal with these situations, i.e., in this case, those below 21 are displayed a message asking them to leave and those above 21 are allowed to enter. A custom exception is to create our own kind of yes/ no scenarios.

4) Encapsulation

Encapsulation is the process of hiding methods or variables from a class and using access specifiers such as default, private, protected or public to mention how a method or variable can be accessed from another class.
Typically, all private variables can only be accessed from methods of the class it is declared in and those methods are made public so as to gain access to variables of that class through its class methods.

5) Polymorphism

Suppose an interface declares certain methods/functions. The classes that implement this interface can have different implementations for each method. This is known as polymorphism where one object or method can perform multiple actions.
2 types of polymorphism : Compile-time and Run-time.
In Compile Time polymorphism, method overloading can be done.
In Run Time polymorphism, method overriding can be done

6) Overloading

When multiple methods with the same name but different signatures exist in a class, it is referred to as Method Overloading. When multiple constructors exist with different numbers or types of arguments, it is referred to as Constructor Overloading.

Eg :
```
class Calculate {
        int a=12;
        int b=3;

        public void calculate() {
                System.out.println(a+b);
        }
        public void calculate(int a) {
                this.a=a;
                System.out.println(a+b);
        }
        public void calculate(int a, int b) {
                this.a=a;
                this.b=b;
                System.out.println(a+b);
        }

        public void display() {
                System.out.println("Display func1");
        }

        public void display(int a) {
                System.out.println("Display func2, "+a);
        }

        public void display(int a, int b) {
                System.out.println("Display func3, "+a+" "+b);
        }

}
```

7) Overriding

When a child/derived/subclass has the same method as its parent/base/super class but a different implementation, it is referred to as Method Overriding.

```
class A {
        public void display() {
                System.out.println(“Display classA”);
        }
}
class B extends class A {
        public void display() {
                System.out.println(“Display classB”);
        }
}
```

8) Output

X-workZ

9) Output

Exception - Main method does not exist.

10) Output

X-workz provide java Enterprise application training
Institutes provide training

11) Abstraction

Abstraction is a process used to hide the backend working of a particular program and show only certain details to a user.
Abstract classes or interfaces may be used to achieve this.
Abstract classes are those that contain only abstract methods which must later on be implemented in its subclasses.
Abstract methods are those that have no body but only a declaration in an abstract class. These methods are then implemented in the classes that extend this abstract class.
An abstract class may contain both abstract or concrete methods but all abstract methods must be declared with the abstract keyword in abstract classes only.
A method or class is made abstract with the keyword ‘abstract’.
100% abstraction can be achieved using interfaces which can then be implemented by multiple interfaces or classes.

12) Initializing a value

Usually, initialization of a value is done either using the ‘equal’ operator (=).
Eg : int a =12;
An object can be initialized using either a constructor of that particular class or using the new operator.
Eg :
Object obj = new Object (parameter list as per constructor)
or

```
Object obj = new Object();
```