Question 1.

===========

What Is a Class?

A class is the blueprint (or template or prototype) from which individual objects are created.

Class is kind of a common noun. For example, city.

In the real world, you'll often find many individual objects all of the same kind. There may be thousands of other bicycles in existence, all of the same make and model. Each bicycle was built from the same set of blueprints and therefore contains the same components. In object-oriented terms, we say that your bicycle is an instance of the class of objects known as bicycles

The following Bicycle class is one possible implementation of a bicycle:

```
class Bicycle {

    int cadence = 0;

    int speed = 0;

    int gear = 1;


    void changeCadence(int newValue) {

        cadence = newValue;

    }


    void changeGear(int newValue) {

        gear = newValue;
    }


    void speedUp(int increment) {

         speed = speed + increment;

    }


    void applyBrakes(int decrement) {

         speed = speed - decrement;

    }


    void printStates() {

         System.out.println("cadence:" +

             cadence + " speed:" +

             speed + " gear:" + gear);

    }

}
```

The syntax of the Java programming language will look new to you, but the design of this class is based on the previous discussion of bicycle objects. The fields cadence, speed, and gear represent the object's state, and the methods (changeCadence, changeGear, speedUp etc.) define its interaction with the outside world.

You may have noticed that the Bicycle class does not contain a main method. That's because it's not a complete application; it's just the blueprint for bicycles that might be used in an application. The responsibility of creating and using new Bicycle objects belongs to some other class in your application.

Question 2

===========

Object in Java.

Objec is kind of a proper noun. For example, Toronot, Calgary, Windsor, etc.

An entity that has state and behavior is known as an object e.g. chair, bike, marker, pen, table, car etc. It can be physical or logical (tengible and intengible). The example of integible object is banking system.

An object has three characteristics:

state: represents data (value) of an object.

behavior: represents the behavior (functionality) of an object such as deposit, withdraw etc.

identity: Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But,it is used internally by the JVM to identify each object uniquely.

For Example: Pen is an object. Its name is Reynolds, color is white etc. known as its state. It is used to write, so writing is its behavior.

Object is an instance of a class. Class is a template or blueprint from which objects are created. So object is the instance(result) of a class.

Question 3

==========

An instance is a unique copy of a Class that representing an Object. When a new instance of a class is created, the JVM will allocate a room of memory for that class instance.

Instance is kind of a space allocation of the city's (Toronto, Calgary, Windsor) in Canada.

Question 4

===========

A class is kind of the blueprint or template that describes the methods and variables that will be in each object. An object is an instantiated (instance of a) class; an object is something, while the class is simply the plans to make that something.

Class in Java

A class is a group of objects that has common properties. It is a template or blueprint from which objects are created.

A class in java can contain:

- data member
- method
- constructor
- block
- class and interface

Syntax to declare a class:

```
class <class_name>{
    data member;
    method;
}
```

Simple Example of Object and Class

In this example, we have created a Student class that have two data members id and name. We are creating the object of the Student class by new keyword and printing the objects value.

```
class Student1{
  int id;//data member (also instance variable)
  String name;//data member(also instance variable)
```

```
    public static void main(String args[]){
     Student1 s1=new Student1();//creating an object of Student
     System.out.println(s1.id);
     System.out.println(s1.name);
    }
  }
```

Question 5

===========

Anonymous class is the class without having a name tag, hence only one object can be created and one cannot create more than one object as there is no name of the class.

While a regular class has a name and thus can be used to create as many objects of it.

Question 6

===========

An interface in java is a blueprint of a class. It has static constants and abstract methods only.

The interface in java is a mechanism to achieve fully abstraction. There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritance in Java.

Java Interface also represents IS-A relationship.

It cannot be instantiated just like abstract class.

```
    interface Circle {
  public static final BigDecimal PI =    3.1428571
  public BigDecimal areaOfCircle(BigDecimal radius);
```

```
    public BigDecimal circumferenceOfCircle(BigDecimal radius);

        }


        class MyCircle implements Circle{


        public BigDecimal areaOfCircle(BigDecimal radius) {

             return PI.multiply(radius).multiply(radius);

}


public BidDecimal circumferenceOfCircle(BigDecimal radius) {

             return PI.multiply(new (BigDecimal(2)).multiply(radius);

}

}


Class MainClass {

        public static void main(String args[]){

  BigDecimal rad = new BigDecimal(2.5);

    MyCircle circleOne = new MyCircle();


system.out.println("Area :     " +       circleOne .areaOfCircle(rad).toString());

system.out.println("Perimeter :     " +       circleOne .circumferenceOfCircle(rad).toString());

          }

        }
```

Question 7

=============

Class has both definition and an implementation (carries logic) whereas Interface only has a definition (signature) of methods/behaviour.

Interface is kind of headings only (the index page of a book) while Class carries the body of the paragraph (carrying logic) of the headings mentioned in interface (the index page).

Interfaces are actually implemented via a Class.

Interface contains only method prototype only, it doesn't provide implementation to its method, we can provide implemetation to its method in our class for that one we can implement the interface.

interface is the source for polymorphism.

Interface is kind of a question paper, while class is kind of Answers to those question paper and has detail explanation of those questions.

Question 8

============

An anonymous class is a local class without a name. An anonymous class is defined and instantiated in a single succinct expression using the new operator. While a local class definition is a statement in a block of Java code, an anonymous class definition is an expression, which means that it can be included as part of a larger expression, such as a method call. When a local class is used only once, consider using anonymous class syntax, which places the definition and use of the class in exactly the same place.

For example, new Earth(){ ....}

Question 9

===========

Anonymous class can be defined starting a key word "new". and is used for when do not need a second or more objects of it. It is kind of a local inline class.

Since it does not have any name, therefore it gets instantiated only once, plays the role define in it and vanish for ever.

That is , it is a one time process per execution of the parent block/method.

Question 10

===============

Concrete class is template and one can produce as many objecs as he wants by using the specified name of the class, and it can be an independant class.

While Anonymous class depends on the execution of its parent block/method and gets only one chance per execution of its parent

To instantiate of concrete class, once call by its name and allocate a memory of its object through "new" keyword. while if one can instantiate an anonyouse class by executing of its parent block/method.