1(a) Write a JAVA program to implement class mechanism.

Program:

```
class Student {
   String name;
   int rollNumber;

   void setDetails(String name, int rollNumber) {
      this.name = name;
      this.rollNumber = rollNumber;
   }

   void displayDetails() {
      System.out.println("Name: " + name);
      System.out.println("Roll Number: " + rollNumber);
   }
}

public class Main {
   public static void main(String[] args) {
      Student student = new Student();
      student.setDetails("Ahamed", 101);
      student.displayDetails();
   }
}
```

Output:

```
Name: Ahamed
Roll Number: 101
```

1(b) Write a JAVA program to search for an element using binary search.

Program:

```
import java.util.Arrays;
import java.util.Scanner;

public class BinarySearch {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int[] arr = {10, 20, 30, 40, 50};
      Arrays.sort(arr);
```

```
        System.out.println("Enter element to search: ");
        int key = sc.nextInt();

        int result = Arrays.binarySearch(arr, key);

        if (result >= 0) {
            System.out.println("Element found at index: " + result);
        } else {
            System.out.println("Element not found.");
        }
    }
}
```

Output Example:

```
Enter element to search:
30
Element found at index: 2
```

2(a) Write a JAVA program to implement method overloading.

Program:

```
class Calculator {
    int add(int a, int b) {
        return a + b;
    }

    double add(double a, double b) {
        return a + b;
    }
}

public class Main {
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("Sum (int): " + calc.add(10, 20));
        System.out.println("Sum (double): " + calc.add(10.5, 20.5));
    }
}
```

Output:

```
Sum (int): 30
Sum (double): 31.0
```

2(b) Write a JAVA program to implement Bubble Sort.

Program:

```
public class BubbleSort {
    public static void main(String[] args) {
        int[] arr = {64, 34, 25, 12, 22, 11, 90};
        int n = arr.length;

        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    // Swap arr[j] and arr[j+1]
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }

        System.out.println("Sorted array:");
        for (int value : arr) {
            System.out.print(value + " ");
        }
    }
}
```

Output:

```
Sorted array:
11 12 22 25 34 64 90
```

Let me know which other programs you'd like me to elaborate on!

3(a) Write a JAVA program to implement a constructor.

Program:

```
class Person {
    String name;
    int age;
    // Constructor
    Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    void display() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

public class Main {
    public static void main(String[] args) {
        Person person = new Person("Ahamed", 21);
        person.display();
    }
}
```

Output:

```
Name: Ahamed
Age: 21
```

3(b) Write a JAVA program using StringBuffer to delete and remove a character.

Program:

```
public class StringBufferExample {
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Hello World");

        System.out.println("Original String: " + sb);

        // Delete a portion of the string
        sb.delete(5, 11);
        System.out.println("After deletion: " + sb);

        // Remove a single character
        sb.deleteCharAt(0);
        System.out.println("After removing character: " + sb);
    }
}
```

Output:

```
Original String: Hello World
After deletion: Hello
After removing character: ello
```

4(a) Write a JAVA program to implement constructor overloading.

Program:

```
class Rectangle {
   int length, width;

   // Constructor 1
   Rectangle() {
      length = 0;
      width = 0;
   }

   // Constructor 2
   Rectangle(int side) {
      length = width = side;
   }

   // Constructor 3
   Rectangle(int length, int width) {
      this.length = length;
      this.width = width;
   }

   int area() {
      return length * width;
   }
}

public class Main {
   public static void main(String[] args) {
      Rectangle rect1 = new Rectangle();
      Rectangle rect2 = new Rectangle(5);
      Rectangle rect3 = new Rectangle(4, 6);

      System.out.println("Area of default rectangle: " + rect1.area());
      System.out.println("Area of square: " + rect2.area());
      System.out.println("Area of rectangle: " + rect3.area());
   }
}
```

Output:

```
Area of default rectangle: 0
Area of square: 25
Area of rectangle: 24
```

4(b) Write a JAVA program to import and use user-defined packages.

Program:

1. Create a package mypackage:

```
// Save this as Calculator.java in folder mypackage
package mypackage;

public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }

    public int subtract(int a, int b) {
        return a - b;
    }
}
```

2. Use the package in a program:

```
import mypackage.Calculator;

public class Main {
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("Sum: " + calc.add(10, 20));
        System.out.println("Difference: " + calc.subtract(20, 10));
    }
}
```

Output:

```
Sum: 30
Difference: 10
```

5(a) Write a JAVA program to implement Single Inheritance.

Program:

```
class Animal {
    void eat() {
        System.out.println("This animal eats food.");
    }
}

class Dog extends Animal {
    void bark() {
        System.out.println("The dog barks.");
    }
}

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat();
        dog.bark();
    }
}
```

Output:

```
This animal eats food.
The dog barks.
```

5(b) Write a JAVA program to create threads by extending the Thread class.

Program:

```
class Thread1 extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("Good Morning");
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}

class Thread2 extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("Hello");
                Thread.sleep(2000);
            }
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}

class Thread3 extends Thread {
    public void run() {
        try {
            while (true) {
                System.out.println("Welcome");
                Thread.sleep(3000);
            }
        } catch (InterruptedException e) {
            System.out.println(e);
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Thread1 t1 = new Thread1();
        Thread2 t2 = new Thread2();
        Thread3 t3 = new Thread3();

        t1.start();
        t2.start();
        t3.start();
    }
}
```

Output (Repeated every few seconds):

```
Good Morning
```

```
Hello
Good Morning
Welcome
Good Morning
Hello
Good Morning
...
```

Would you like more examples for the other points?

6(a) Write a JAVA program to implement multi-level inheritance.

Program:

```java
class Animal {
    void eat() {
        System.out.println("Animals eat food.");
    }
}

class Mammal extends Animal {
    void walk() {
        System.out.println("Mammals walk.");
    }
}

class Dog extends Mammal {
    void bark() {
        System.out.println("Dogs bark.");
    }
}

public class Main {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.eat();
        dog.walk();
        dog.bark();
    }
}
```

Output:

```
Animals eat food.
Mammals walk.
```

Dogs bark.

6(b) Write a JAVA program for an abstract class to find areas of different shapes.

Program:

```
abstract class Shape {
    abstract void area();
}

class Circle extends Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    void area() {
        System.out.println("Area of Circle: " + (Math.PI * radius * radius));
    }
}

class Rectangle extends Shape {
    double length, width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    void area() {
        System.out.println("Area of Rectangle: " + (length * width));
    }
}

public class Main {
    public static void main(String[] args) {
        Shape circle = new Circle(5);
        circle.area();

        Shape rectangle = new Rectangle(4, 6);
        rectangle.area();
    }
}
```

Output:

```
Area of Circle: 78.53981633974483
Area of Rectangle: 24.0
```

7(a) Write a JAVA program to give an example for the super keyword.

Program:

```java
class Animal {
   String name = "Animal";

   void sound() {
      System.out.println("Animal makes a sound.");
   }
}

class Dog extends Animal {
   String name = "Dog";

   void displayNames() {
      System.out.println("Child class name: " + name);
      System.out.println("Parent class name: " + super.name);
   }

   void sound() {
      super.sound(); // Call parent method
      System.out.println("Dog barks.");
   }
}

public class Main {
   public static void main(String[] args) {
      Dog dog = new Dog();
      dog.displayNames();
      dog.sound();
   }
}
```

Output:

```
Child class name: Dog
Parent class name: Animal
Animal makes a sound.
Dog barks.
```

7(b) Write a JAVA program to implement an interface.

Program:

```
interface Animal {
    void eat();
}

class Dog implements Animal {
    public void eat() {
        System.out.println("Dog eats bones.");
    }
}

class Cat implements Animal {
    public void eat() {
        System.out.println("Cat eats fish.");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal dog = new Dog();
        dog.eat();

        Animal cat = new Cat();
        cat.eat();
    }
}
```

Output:

```
Dog eats bones.
Cat eats fish.
```

8(a) Write a JAVA program that implements runtime polymorphism.

Program:

```
class Animal {
    void sound() {
        System.out.println("Animals make sounds.");
    }
}

class Dog extends Animal {
    void sound() {
        System.out.println("Dog barks.");
    }
}

class Cat extends Animal {
    void sound() {
        System.out.println("Cat meows.");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal animal;
        animal = new Dog();
        animal.sound();

        animal = new Cat();
        animal.sound();
    }
}
```

Output:

```
Dog barks.
Cat meows.
```

8(b) Write a JAVA program illustrating multiple catch clauses.

Program:

```
public class MultipleCatch {
    public static void main(String[] args) {
        try {
            int[] arr = new int[5];
            arr[10] = 30 / 0;
        } catch (ArithmeticException e) {
            System.out.println("ArithmeticException: Division by zero.");
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("ArrayIndexOutOfBoundsException: Invalid index access.");
        } catch (Exception e) {
            System.out.println("General Exception: " + e);
        }
    }
}
```

Output:

```
ArithmeticException: Division by zero.
```

Would you like more examples for other points?

9(a) Write a JAVA program that describes the exception handling mechanism.

Program:

```
public class ExceptionHandling {
    public static void main(String[] args) {
        try {
            int a = 10, b = 0;
            int result = a / b; // This will throw an exception
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("ArithmeticException: Division by zero is not allowed.");
        } finally {
            System.out.println("Finally block executed.");
        }
    }
}
```

Output:

```
ArithmeticException: Division by zero is not allowed.
Finally block executed.
```

9(b) Write a JAVA program to give an example for the super keyword.

Program:

```
class Vehicle {
    int speed = 50;
    void display() {
        System.out.println("Vehicle speed: " + speed);
    }
}

class Car extends Vehicle {
    int speed = 100;

    void display() {
        super.display(); // Call parent class method
        System.out.println("Car speed: " + speed);
    }
}

public class Main {
    public static void main(String[] args) {
        Car car = new Car();
        car.display();
    }
}
```

Output:

```
Vehicle speed: 50
Car speed: 100
```

10(a) Write a JAVA program illustrating multiple catch clauses.

Program:

```
public class MultipleCatchExample {
    public static void main(String[] args) {
        try {
            String str = null;
            System.out.println(str.length()); // This will throw a NullPointerException
        } catch (ArithmeticException e) {
            System.out.println("Arithmetic Exception occurred.");
        } catch (NullPointerException e) {
            System.out.println("NullPointerException occurred.");
        } catch (Exception e) {
            System.out.println("An unknown exception occurred: " + e);
        }
    }
}
```

Output:

```
NullPointerException occurred.
```

10(b) Write a JAVA program to display default values of all primitive data types in JAVA.

Program:

```java
class DefaultValues {
    byte b;
    short s;
    int i;
    long l;
    float f;
    double d;
    char c;
    boolean bool;

    void displayDefaults() {
        System.out.println("Default byte: " + b);
        System.out.println("Default short: " + s);
        System.out.println("Default int: " + i);
        System.out.println("Default long: " + l);
        System.out.println("Default float: " + f);
        System.out.println("Default double: " + d);
        System.out.println("Default char: [" + c + "]");
        System.out.println("Default boolean: " + bool);
    }
}

public class Main {
    public static void main(String[] args) {
        DefaultValues defaults = new DefaultValues();
        defaults.displayDefaults();
    }
}
```

Output:

```
Default byte: 0
Default short: 0
Default int: 0
Default long: 0
Default float: 0.0
Default double: 0.0
Default char: []
Default boolean: false
```

11(a) Write a JAVA program for creation of Java Built-in Exceptions.

Program:

```
public class BuiltInException {
    public static void main(String[] args) {
        try {
            int[] arr = {1, 2, 3};
            System.out.println(arr[5]); // This will throw ArrayIndexOutOfBoundsException
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Caught an ArrayIndexOutOfBoundsException: " +
e.getMessage());
        }
    }
}
```

Output:

```
Caught an ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 3
```

11(b) Write a JAVA program to implement multi-level inheritance.

Program:

```
class Animal {
    void sound() {
        System.out.println("Animals make sound.");
    }
}

class Mammal extends Animal {
    void walk() {
        System.out.println("Mammals can walk.");
    }
}

class Human extends Mammal {
    void speak() {
        System.out.println("Humans can speak.");
    }
}

public class Main {
    public static void main(String[] args) {
        Human human = new Human();
        human.sound();
        human.walk();
        human.speak();
    }
}
```

Output:

```
Animals make sound.
Mammals can walk.
Humans can speak.
```

12(a) Write a JAVA program for creation of User Defined Exception.

Program:

```
class MyException extends Exception {
    public MyException(String message) {
        super(message);
    }
}

public class UserDefinedException {
    public static void main(String[] args) {
        try {
            throw new MyException("This is a user-defined exception.");
        } catch (MyException e) {
            System.out.println("Caught Exception: " + e.getMessage());
        }
    }
}
```

Output:

```
Caught Exception: This is a user-defined exception.
```

12(b) Write a JAVA program for an abstract class to find areas of different shapes.

Program:

```java
abstract class Shape {
    abstract void area();
}

class Triangle extends Shape {
    double base, height;

    Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    void area() {
        System.out.println("Area of Triangle: " + (0.5 * base * height));
    }
}

class Square extends Shape {
    double side;

    Square(double side) {
        this.side = side;
    }

    void area() {
        System.out.println("Area of Square: " + (side * side));
    }
}

public class Main {
    public static void main(String[] args) {
        Shape triangle = new Triangle(5, 10);
        triangle.area();

        Shape square = new Square(4);
        square.area();
    }
}
```

Output:

```
Area of Triangle: 25.0
Area of Square: 16.0
```

Let me know if you need further assistance!