[[[1]]] //Time

```
public class Time {
        public int Hour ;
        public int Minute ;
        public int Second ;
        public long Time ;

        public Time() {
                Time = System.currentTimeMillis();
        }
        public Time(long time) {
                Time = time ;
        }
        public Time(int hour, int minute, int second) {
                Hour = hour;
                Minute = minute;
                Second = second;
        }
        public int getHour() {
                return (int)(Time / (1000 * 60 * 60)) % 24 ;
        }
        public int getMinute() {
                return (int)(Time / (1000 * 60)) % 60 ;
        }
        public int getSecond() {
                return (int)(Time / 1000) % 60 ;
        }
        public void setTime (long elaspeTime) {
                Time = elaspeTime ;
        }
}

public class TestTime {
        public static void main(String [] args) {

                Time time1 = new Time () ;
                Time time2 = new Time (555550000);

                System.out.println("For time 1 : \nHour
:"+time1.getHour()+"\nMinutes : "+time1.getMinute()+"\nSeconds :
"+time1.getSecond());
                System.out.println("\n\nFor time 2 : \nHour
:"+time2.getHour()+"\nMinutes : "+time2.getMinute()+"\nSeconds :
"+time2.getSecond());

        }

}
```

[[[2]]] // MyInteger

```
public class MyInteger {
	private int value ;

	public MyInteger(int value) {
		this.value = value;
	}
	public int getValue() {
		return value;
	}
	public boolean isEven() {
		return (value % 2) == 0 ;
	}
	public boolean isOdd() {
		return (value % 2) != 0 ;
	}
	public boolean isPrime() {
		if(value == 1 || value == 2) {
			return true ;
		}
		else {
		      for (int i = 2 ; i < value ; i++) {
		          if(value % i == 0) {
		                    return false ;
		          }
		      }	 return true ;
		}
	}
	public static boolean isEven(int newValue) {
		return (newValue % 2) == 0 ;
	}
	public static boolean isOdd(int newValue) {
		return (newValue % 2) != 0 ;
	}
	public static boolean isPrime(int newValue) {
		if(newValue == 1 || newValue == 2) {
			return true ;
		}
		else {
		      for (int i = 2 ; i < newValue ; i++) {
		          if(newValue % i == 0) {
		                    return false ;
		          }
		      }  return true ;
		}
	}
	public boolean isEven(MyInteger value) {
		return value.isEven() ;
	}
	public boolean isOdd(MyInteger value) {
```

```
                return value.isOdd() ;
        }
        public boolean isPrime(MyInteger value) {
                return value.isPrime();
        }
        public boolean equals(int value) {
                if(this.value == value) {
                        return true ;
                } return false ;
        }
        public boolean equals(MyInteger newValue) {
                if(newValue.value == this.value) {
                        return true ;
                } return false ;
        }
        public static int parseInt(char [] value) {
                int sum = 0 ;
                for(int i = 0 ; i < value.length ; i++) {
                        sum = sum * 10 + Character.getNumericValue(value[i]);
                } return sum ;
        }
        public static int parseInt(String value) {
                return Integer.parseInt(value);
        }
}
public class TestMyInteger {

        public static void main(String[] args) {
                MyInteger int1 = new MyInteger(23);
        MyInteger int2 = new MyInteger(56);
        MyInteger int3 = new MyInteger(2);
        MyInteger int4 = new MyInteger(23);

        System.out.printf("%d is prime? %s%n", int1.getValue(), int1.isPrime());
        System.out.printf("%d is prime? %s%n", int2.getValue(), int2.isPrime());
        System.out.printf("%d is prime? %s%n", int3.getValue(), int3.isPrime());

        System.out.printf("%d is even? %s%n", int1.getValue(), int1.isEven());
        System.out.printf("%d is even? %s%n", int2.getValue(), int2.isEven());
        System.out.printf("%d is even? %s%n", int3.getValue(), int3.isEven());

        System.out.printf("93 is odd? %s%n", MyInteger.isOdd(93));

        System.out.printf("%d equals %d? %s%n", int1.getValue(), int4.getValue(),
int1.equals(int4));

        System.out.printf("%d%n", MyInteger.parseInt(new char[] {'1', '5', '6'}));
        System.out.printf("%d%n", MyInteger.parseInt("454"));

        }
```

```
}
[[[3]]]  //Queue

public class Queue {

    private int [] elements ;
    private int size ;

    public Queue() {
                elements = new int [8] ; //default value 8
        }
    public void enqueue(int v) {
        if(size >= elements.length) {
                int [] temp = new int [elements.length * 2] ; //length doubled as
per the condition
                System.arraycopy(elements, 0, temp, 0, elements.length);
                elements = temp ;
        }
        elements[size++] = v ;
    }
    public int dequeue () {
        int v = elements[0] ;
        size-- ;
        for(int i = 0 ; i < size ; i++) {
                elements[i] = elements[i+1] ;
        }
        return v ;
    }
    public boolean isEmpty() {
        return size == 0 ;
    }
    public int getSize() {
        return size ;
    }
}
public class TestQueue {
        public static void main(String[] args) {

                Queue queue = new Queue () ;
                //inserting 20 numbers
                for(int i = 1 ; i <= 20 ; i++) {
                        queue.enqueue(i);
                }
                //displaying 20 numbers
                for(int i = 0 ; i < 20 ; i++) {
                        System.out.print(queue.dequeue()+" ");
                }
        }
}
```

[[[4]]] //Circle2D

```
public class Circle2D {
	private double x ;
	private double y ;
	private double radius ;

	public Circle2D() {
		x = 0 ;
		y = 0 ;
		radius = 1 ;
	}
	public Circle2D(double x, double y, double radius) {
		this.x = x;
		this.y = y;
		this.radius = radius;
	}
	public double getX() {
		return x;
	}
	public double getY() {
		return y;
	}
	public double getRadius() {
		return radius;
	}
	public double getArea() {
		return Math.PI * radius * radius ;
	}
	public double getPerimeter() {
		return 2* Math.PI * radius ;
	}
	public boolean contains(double x , double y) {
		if(Math.sqrt(Math.pow(this.x - x , 2) + Math.pow(this.y - y , 2)) <
radius) {
			return true ;
		} return false ;
	}
	public boolean contains(Circle2D circle) {
		if(Math.sqrt(Math.pow(this.x - circle.x , 2) + Math.pow(this.y -
circle.y , 2)) + circle.radius <= radius) {
			return true ;
		} return false ;
	}
	public boolean overlaps(Circle2D circle) {
		if(Math.sqrt(Math.pow(this.x - circle.x , 2) + Math.pow(this.y -
circle.y , 2)) <= radius + circle.radius) {
			return true ;
		} return false ;
	}
```

```
}
public class TestCircle2D {
        public static void main(String[] args) {

                Circle2D c1 = new Circle2D(2, 2, 5.5);
                System.out.println("Area : " +c1.getArea());
                System.out.println("Perimeter : " +c1.getPerimeter());
                System.out.println("Contains point : " +c1.contains(3,3));
                System.out.println("Contains point : " +c1.contains(new
Circle2D(4,5,10.5)));
                System.out.println("Overlaps point : " +c1.contains(new
Circle2D(3,5,2.3)));

        }
}
```

[[[5]]]

```
public class MyRectangle2D {
    private double x , y ;
    private double width , height ;

    public MyRectangle2D(){
        x = y = 0 ;
        width = height = 1 ;
    }
    public MyRectangle2D(double x , double y,double width,double height){
        this.x = x ;
        this.y = y ;
        this.width = width ;
        this.height = height ;
    }
    public double getPerimeter (){
        return 2*(width+height);
    }
    public double getArea(){
        return width * height ;
    }
    public boolean contains (double x, double y){
        return Math.abs(x-this.x) <= width/2 && Math.abs(y-this.y) <= height/2 ;
    }
    public boolean contains (MyRectangle2D r){
        return (contains(r.x -r.width /2 , r.y+r.height /2) &&
                contains(r.x -r.width /2 , r.y-r.height /2) &&
                contains(r.x +r.width /2 , r.y+r.height /2) &&
                contains(r.x +r.width /2 , r.y-r.height /2)) ;
    }
     public boolean overlaps (MyRectangle2D r){
        return Math.abs(r.x-x) <= (r.width+width)/2 && Math.abs(r.y-y)
```

```
<=(r.height+height)/2 ;
    }
}

public class TestMyRectangle{
        public static void main(String[] args){
            MyRectangle2D rect = new MyRectangle2D(2,2,5.5,4.9);

            System.out.println("Area is "+rect.getArea());
            System.out.println("Perimeter is "+rect.getPerimeter());
            System.out.println("Contains (3,3) "+rect.contains(3,3));
            System.out.println("Contains (4,5,10.5,3.2) "+rect.contains(new
MyRectangle2D(4,6,10.5,3.2)));
            System.out.println("Overlaps "+rect.overlaps(new
MyRectangle2D(3,5,2.3,6.7)));
      }
 }
```

[[[6]]]

```
import java.util.Calendar ;
import java.util.GregorianCalendar ; //importing GregorianCalender class

public class MyDate {
        private int year ;
        private int month ;
        private int day ;

        public MyDate() {
                GregorianCalendar calendar = new GregorianCalendar();
                year = calendar.get(GregorianCalendar.YEAR);
                month = calendar.get(GregorianCalendar.MONTH);
                day = calendar.get(GregorianCalendar.DAY_OF_MONTH);
        }
        public MyDate(long elaspedTime) {
                GregorianCalendar calendar = new GregorianCalendar();
                calendar.setTimeInMillis(elaspedTime);
                year = calendar.get(GregorianCalendar.YEAR);
                month = calendar.get(GregorianCalendar.MONTH);
                day = calendar.get(GregorianCalendar.DAY_OF_MONTH);
        }
        public MyDate(int year, int month, int day) {
                this.year = year;
                this.month = month;
                this.day = day;
        }
        public int getYear() {
                return year;
        }
        public int getMonth() {
```

```
                return month;
        }
        public int getDay() {
                return day;
        }
        public void setDate(long elaspedTime) {
                GregorianCalendar calendar = new GregorianCalendar();
                calendar.setTimeInMillis(elaspedTime);
                year = calendar.get(GregorianCalendar.YEAR);
                month = calendar.get(GregorianCalendar.MONTH);
                day = calendar.get(GregorianCalendar.DAY_OF_MONTH);
        }

}

public class TestMyDate {
        public static void main(String[] args) {

                MyDate date1 = new MyDate();
                MyDate date2 = new MyDate(34355555133101L);

                System.out.println("Date1 - Month : "+date1.getMonth()+" Day :
"+date1.getDay()+" Year : "+date1.getYear());
                System.out.println("Date2 - Month : "+date2.getMonth()+" Day :
"+date2.getDay()+" Year : "+date2.getYear());
        }
}
```