Chapter 3 – Unit 1

Date Class

- The Date class represents date and time in java.
- The Date class represents a specific instant in time, with millisecond precision.
- The Date class available in java.util package, this class encapsulates the current date and time.

java.util.Date Constructors:

No.	Constructor	Description
1)	Date()	Creates a date object representing current date and time.
2)	Date(long milliseconds)	Creates a date object for the given milliseconds since January 1, 1970, 00:00:00 GMT.

java.util.Date Methods:

No.	Method	Description
1)	booleanafter(Date date)	tests if current date is after the given date.
2)	booleanbefore(Date date)	tests if current date is before the given date.
3)	Object clone()	returns the clone object of current date.

4)	int compareTo(Date date)	compares current date with given date.
5)	booleanequals(Date date)	compares current date with given date for equality.
6)	static Date from(Instant instant)	returns an instance of Date object from Instant date.
7)	long getTime()	returns the time represented by this date object.
8)	int hashCode()	returns the hash code value for this date object.
9)	void setTime(long time)	changes the current date and time to given time.
10)	Instant toInstant()	converts current date into Instant object.
11)	String toString()	converts this date into Instant object.

Example: // Show date and time using only Date methods.

```
import java.util.Date;

class dateexample

{

public static void main(String args[])

{

Date date=new Date();

System.out.println(date);
long millis=System.currentTimeMillis();
System.out.println( “Milliseconds since Jan. 1, 1970 GMT =”+millis);
}
}
```

Output

```
Sat Nov 27 14:21:26 IST 2021
Milliseconds since Jan. 1, 1970 GMT = 1638003086091
```

Note : currentTimeMillis(); method returns the current time in milliseconds.

Calendar class

- Calendar class in Java is an abstract class.
- Calendar class provides methods forconverting date between a specific instant in **time** and **a set of calendar fields** such as MONTH, YEAR, HOUR, etc.
- It inherits Object class and implements the Comparable, Serializable, Cloneable interfaces.
- As it is an Abstract class, so we cannot use a constructor to create an instance.
- Instead, we will have to use the static methodCalendar.getInstance() to instantiate and implement a sub-class.

- Calendar.getInstance(): **return a Calendar instance based on the current time in the default time zone with the default locale. (or) To get the instance of calendar according to current time zone set by java Runtime environment.**
- Calendar.getInstance(TimeZone zone)
- Calendar.getInstance(Locale aLocale)
- Calendar.getInstance(TimeZone zone, Locale aLocale)

Example : // Date getTime(): It is used to return aDate object representingthisCalendar's time value.

```
import java.util.*;
class Calendar1
{
public static void main(String args[])
{
Calendar c = Calendar.getInstance();
System.out.println("The Current Date is:" + c.getTime());
}
}
```

Output:

```
The Current Date is: SunNov 28 11:10:40 IST 2021
```

Important Methods and their usage

METHOD	DESCRIPTION
abstract void add(int field, int amount)	It is used to add or subtract the specified amount of time to the given calendar field, based on the calendar’s rules.
int get(int field)	It is used to return the value of the given calendar field.
abstract int getMaximum(int field)	It is used to return the maximum value for the given calendar field of this Calendar instance.
abstract int getMinimum(int field)	It is used to return the minimum value for the given calendar field of this Calendar instance.
Date getTime()	It is used to return a Date object representing this Calendar’s time value.</td

Program 1: Java program to demonstrate get() method.

```
// Program to demonstrate get() methodof Calendar class
import java.util.*;
class Calendar2
{
public static void main(String[] args)
{
// creating Calendar object
Calendar calendar = Calendar.getInstance();
// Demonstrate Calendar's get()method
System.out.println("Current Calendar's Year: " +
calendar.get(Calendar.YEAR));
System.out.println("Current Calendar's Day: " +
calendar.get(Calendar.DATE));
System.out.println("Current MINUTE: " +
calendar.get(Calendar.MINUTE));
System.out.println("Current SECOND: " +
calendar.get(Calendar.SECOND));
}
```

Output:

```
Current Calendar's Year: 2021
Current Calendar's Day: 28
Current MINUTE: 10
Current SECOND: 45
```

Program 2: Java program to demonstrate getMaximum()method.

// Program to demonstrate getMaximum() methodof Calendar class.

```
import java.util.*;
class Calendar3
{
public static void main(String[] args)
{
// creating calendar object
Calendar calendar = Calendar.getInstance();
int max = calendar.getMaximum(Calendar.DAY_OF_WEEK);
System.out.println("Maximum number of days in a week: " +
max);
max = calendar.getMaximum(Calendar.WEEK_OF_YEAR);
System.out.println("Maximum number of weeks in a year: " +
max);
}
}
```

Output:

```
Maximum number of days in a week: 7
Maximum number of weeks in a year: 53
```

Program 3: Java program to demonstrate the getMinimum() method.

// Program to demonstrate getMinimum() methodof Calendar class

```
import java.util.*;
class Calendar4
{
public static void main(String[] args)
{
// creating calendar object
Calendar calendar = Calendar.getInstance();
int min = calendar.getMinimum(Calendar.DAY_OF_WEEK);
System.out.println("Minimum number of days in week: " +
min);
min = calendar.getMinimum(Calendar.WEEK_OF_YEAR);
System.out.println("Minimum number of weeks in year: " +
min);
}
}
```

Output:

```
Minimum number of days in week: 1
Minimum number of weeks in year: 1
```

Program 4: Java program to demonstrate add() method.

// Program to demonstrate add() methodof Calendar class

```
import java.util.*;
class Calendar5
{
public static void main(String[] args)
{
// creating calendar object
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.DATE, -15);
System.out.println("15 days ago: " + calendar.getTime());
calendar.add(Calendar.MONTH, 4);
System.out.println("4 months later: " + calendar.getTime());
calendar.add(Calendar.YEAR, 2);
System.out.println("2 years later: " + calendar.getTime());
}
}
```

Output:

```
15 days ago: Sun Nov 14  10:48:41 IST 2021
4 months later: Mon March 14 10:48:41 IST 2022
2 years later: Thu Mar 14 10:48:41 IST 2024
```

Random class

- Random class is part of java. util package.
- An instance of java Random class is used to generate random numbers.
- This class provides several methods to generate random numbers of type integer, double, long, float etc.
- Java Random class is used to generate a stream of pseudorandom numbers.
- The algorithms implemented by Random class use a protected utility method than can supply up to 32 pseudorandomly generated bits on each invocation.

Methods

Methods	Description
doubles()	Returns an unlimited stream of pseudorandom double values.
ints()	Returns an unlimited stream of pseudorandom int values.
longs()	Returns an unlimited stream of pseudorandom long values.
next()	Generates the next pseudorandom number.
nextBoolean()	Returns the next uniformly distributed pseudorandom boolean value from the random number generator's sequence
nextByte()	Generates random bytes and puts them into a specified byte array.
nextDouble()	Returns the next pseudorandom Double value between 0.0 and 1.0 from the random number generator's sequence
nextFloat()	Returns the next uniformly distributed pseudorandom Float value between 0.0 and 1.0 from this random number generator's sequence
nextGaussian()	Returns the next pseudorandom Gaussian double value with mean 0.0 and standard deviation 1.0 from this random number generator's sequence.
nextInt()	Returns a uniformly distributed pseudorandom int value generated from this random number generator's sequence

nextLong()	Returns the next uniformly distributed pseudorandom long value from the random number generator's sequence.
setSeed()	Sets the seed of this random number generator using a single long seed.

Program:

```
import java.util.Random;

class JavaRandomExample

{

public static void main(String[] args)

{

//create random object

Random random= new Random();


//Returns the next pseudorandom integer value

System.out.println("Random Integer value:"+random.nextInt());


// Returns the next pseudorandom boolean value

boolean val1 = random.nextBoolean();

System.out.println("Random booleanvalue : "+val1);


// Returns the next pseudorandom long value

Long val2 = random.nextLong();

System.out.println("Random Long value : "+val2);}
}
```

Output:

```
Random Integer value : -800049765
Random boolean value : true
Random Long value : 1789244058316846167
```

Scanner Class

- **The Scanner class is used to get user input, and it is found in the java.util package.**
- To use the Scanner class, create an object of the class and use any of the available methods found in the Scanner class documentation.
- **Input Types:**
- In the example above, we used the nextLine() method, which is used to read Strings. To read other types, look at the table below:

Method	**Description**
`nextBoolean()`	Reads a `boolean` value from the user
`nextByte()`	Reads a `byte` value from the user
`nextDouble()`	Reads a `double` value from the user
`nextFloat()`	Reads a `float` value from the user
`nextInt()`	Reads a `int` value from the user
`nextLine()`	Reads a `String` value from the user

`nextLong()`	Reads a `long` value from the user
`nextShort()`	Reads a `short` value from the user

Example :

```
import java.util.Scanner;

class sample

{

public static void main(String[] args)

{

Scanner s = new Scanner(System.in);


System.out.println("Enter name:");

String name = s.nextLine();

System.out.println("Enter age:");

int age =s.nextInt();

System.out.println("Enter salary:");

double salary = s.nextDouble();

System.out.println("Name: " + name);

System.out.println("Age: " + age);

System.out.println("Salary: " + salary);

}

}
```

Output

```
Enter name, age and salary:
Siva Kumar 35 75000
Siva Kumar
35
75000
```