Object Oriented Programming
CS F213

J. Jennifer Ranjani

email: jennifer.ranjani@pilani.bits-pilani.ac.in

Chamber: 6121 B, NAB

Consultation: Appointment by e-mail

BITS Pilani

Pilani Campus

Reading Input from Console

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani

Pilani Campus

Command Line Arguments

Command Line Arguments

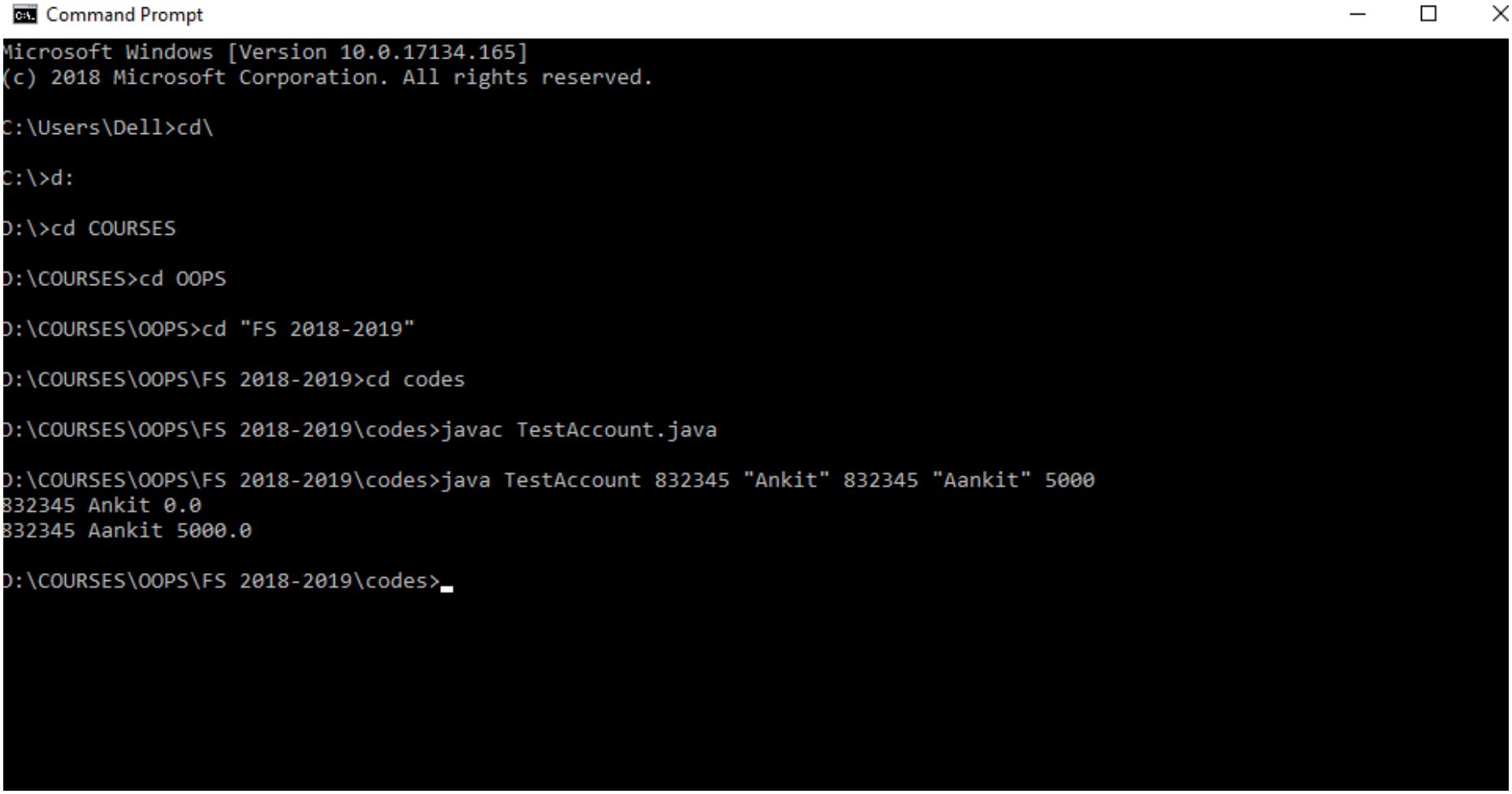
innovate achieve lead

```
class Account{
int acc;
String name;
float amount;
Account(int act,String aname){
acc = act;
name = aname;
}

void update(int act,String aname, float amt) {
acc = act;
name = aname;
amount = amt;
}
```

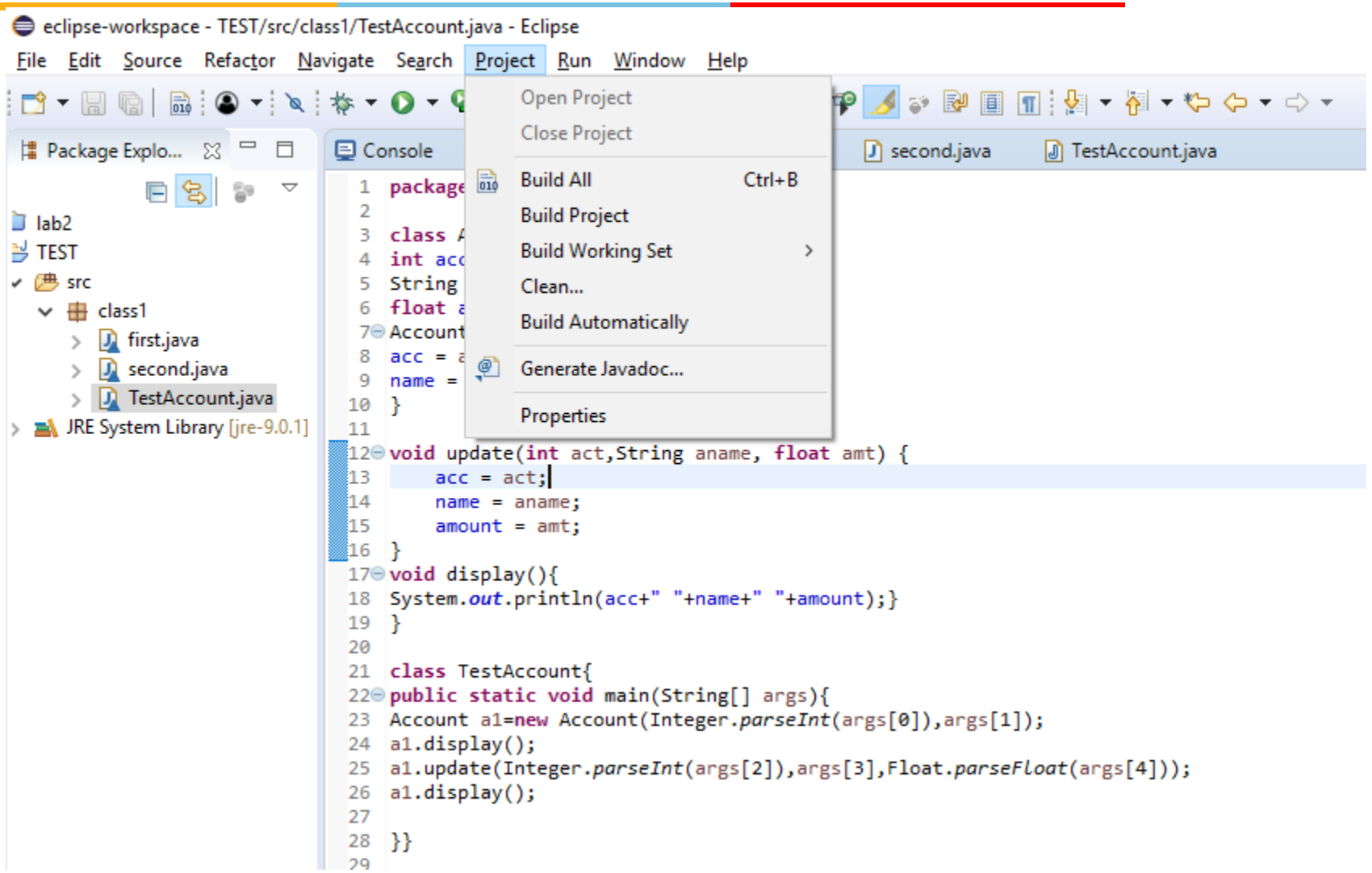
Command Line Arguments

```
void display(){
System.out.println(acc+" "+name+" "+amount);}
}
class second{
public static void main(String[] args){
Account a1=new Account(Integer.parseInt(args[0]),args[1]);
a1.display();
a1.update(Integer.parseInt(args[2]),args[3],Integer.parseInt(args[4]));
a1.display();

}}
```

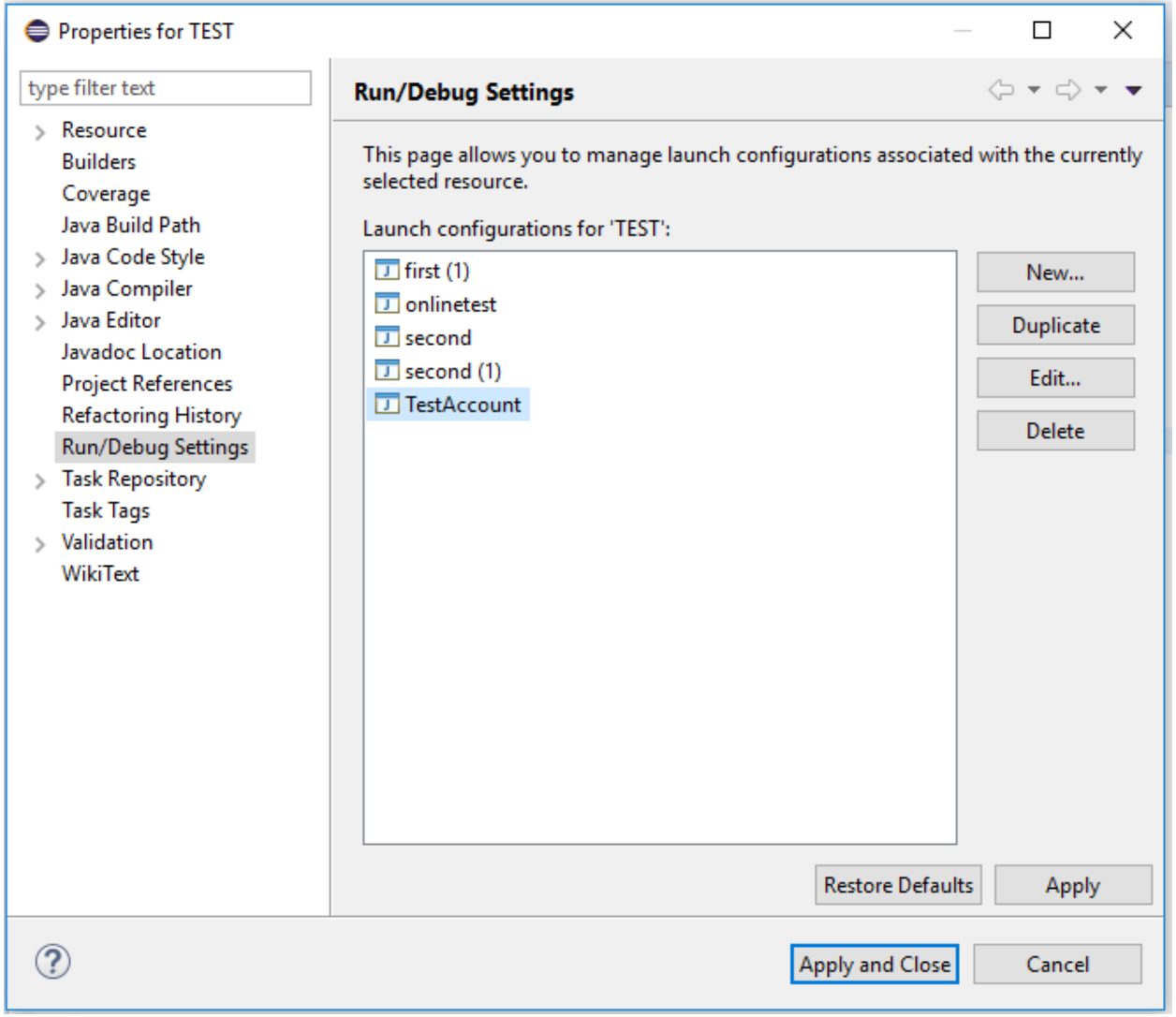
Command Line Arguments (From Command Prompt)



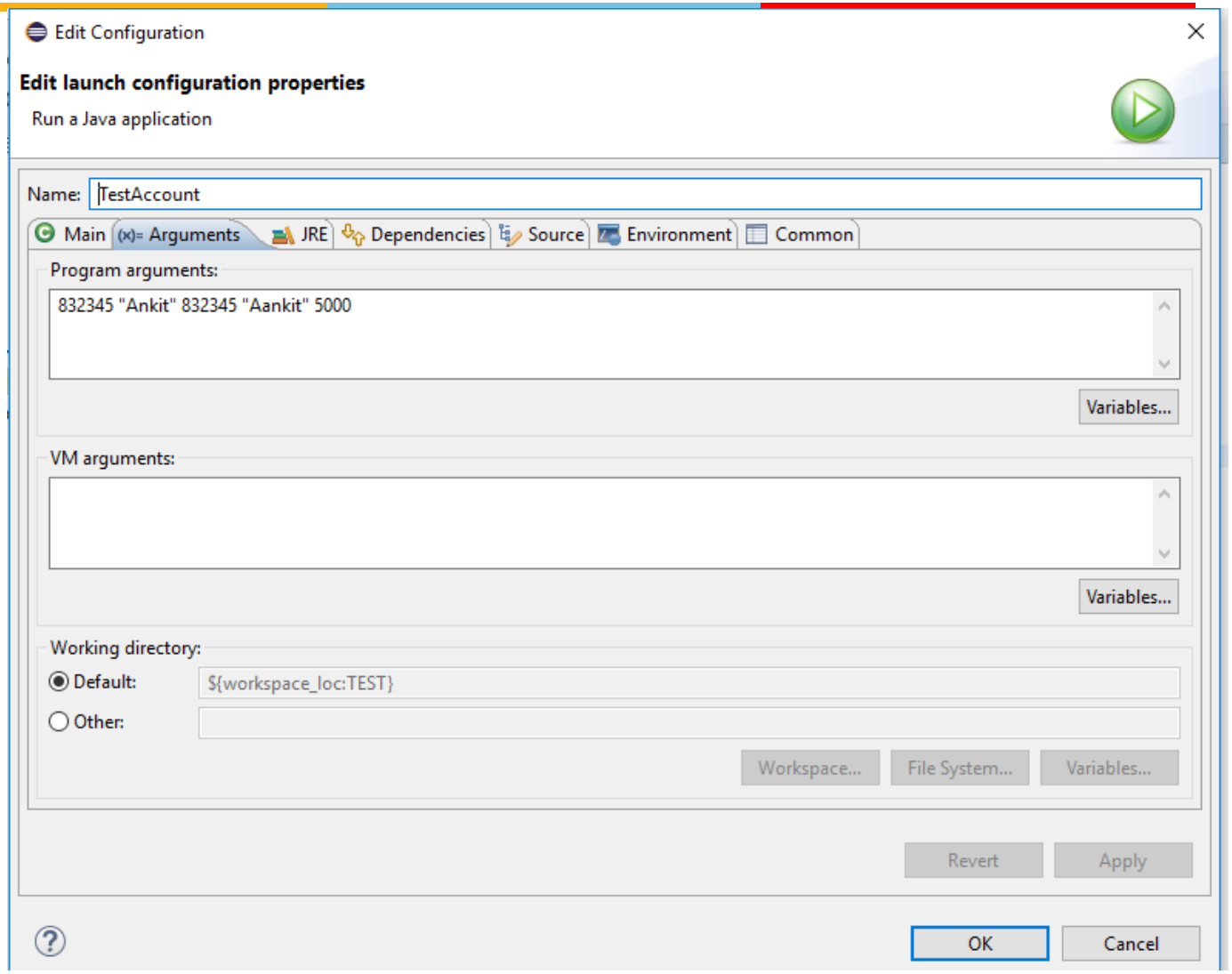
Command Line Arguments (From Eclipse)



Command Line Arguments (From Eclipse)



Command Line Arguments (From Eclipse)



innovate achieve lead

BITS Pilani

Pilani Campus

Array of Objects

Array of Objects

innovate achieve lead

```
class Account{
int acc;
String name;
float amount;
void insert(int acc,String name,float amt){
this.acc = acc;
this.name = name;
this.amount = amt;
}

void display(){
System.out.println(acc+" "+name+" "+amount);}
}
```

Array of Objects

```
class TestAccount{
public static void main(String[] args){
Account[] a= new Account[3];

for(int i=0;i<3;i++)
{
 a[i]= new Account();
 a[i].insert(Integer.parseInt(args[3*i]), args[3*i+1], Float.parseFloat(args[3*i+2]));
a[i].display();
}
}}
```

Output:

111 abc 1000.0
222 bcd 2000.0
333 cde 5000.0

BITS Pilani
Pilani Campus

I/O Streams

Stream

- Java performs I/O using Steams. It is a sequence of data
- In Java, stream is composed of bytes
- Three sources for receiving input and sending output
  - Console I/O, File I/O and Network I/O
- Automatically created streams
  - System.out: Standard output stream
  - System.in: Standard input stream
  - System.err: Standard error stream

Streams Supported by JAVA

- **Byte Stream:** handle I/O of raw binary data; data is 8-bits
- **Character Stream:** Java uses a Unicode system which is a universal international standard character encoding that is capable of representing most of world's written languages. In Unicode, character holds 2 bytes.
- **Buffered Stream**: optimized to reduce the number of I/O calls. It reads a stream of data from memory to a buffer. Native input API is called only when the buffer is empty.
- **Data Stream**: handle binary I/O of primitive data type
- **Object Stream**: handle binary I/O of objets.

BufferedReader

- Reads text from character input stream
- It is advisable to wrap BufferedReader around any Reader whose read() operations are costlier
  - Eg.: BufferedReader in = new BufferedReader(new FileReader("foo.in"));
- Without buffering, each invocation of read() could cause bytes to be read from the file, converted into characters and then its returned
- read() – reads a character or character array of known length from the stream; readLine() – reads a line of text followed by ‘\n’ or \r’

BufferedReader-Example

innovate achieve lead

```
import java.io.*;

class Account{
int acc;
String name;
float amount;

void insert(int acc,String name,float amt){
this.acc = acc;
this.name = name;
this.amount = amt;   }

void display(){
System.out.println(acc+" "+name+" "+amount);}
}
```

```
class TestAccount{
public static void main(String[] args) throws IOException{

Account a= new Account();
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the account no");
int acc ;
acc = Integer.parseInt(br.readLine());

System.out.println("Enter the name");
String name="";
 name = br.readLine();

System.out.println("Enter the amount");
Float amount;
amount = Float.parseFloat(br.readLine());

a.insert(acc, name, amount);
a.display();   }}
```

Console:

```
Enter the account no
111
Enter the name
Ankit
Enter the amount
5000
111 Ankit 5000.0
```

Scanner Class

- It is in java.util package for obtaining input of primitive types like int, double etc. and strings
- Pass an object of System.in or file to create an object of the scanner class.
- To read numerical values of certain data type XYZ, use nextXYZ(). Eg. nextInt(), nextFloat() etc.
- To read strings, use nextLine()
- To read a character array, use next(). It is terminated by space, new line or carriage return
- To read single character, use next().charAt(0)

```
class TestAccount{
public static void main(String[] args) {
Account a= new Account();
Scanner sr = new Scanner(System.in);

System.out.println("Enter the account no");
int acc ;
acc = sr.nextInt();

System.out.println("Enter the name");
String name;
 name = sr.next();

System.out.println("Enter the amount");
Float amount;
amount = sr.nextFloat();

a.insert(acc, name, amount);
a.display();
sr.close();   }}
```

Console:

```
Enter the account no
111
Enter the name
Ankit
Enter the amount
5000
111 Ankit 5000.0
```

```
class TestAccount{
public static void main(String[] args) {
Account a= new Account();
Scanner sr = new Scanner(System.in);

System.out.println("Enter the account no");
int acc ;
acc = sr.nextInt();

System.out.println("Enter the name");
String name;
 name = sr.next();

System.out.println("Enter the amount");
Float amount;
amount = sr.nextFloat();

a.insert(acc, name, amount);
a.display();
sr.close();   }}
```

Console:

```
Enter the account no
111
Enter the name
Ankit Tiwari
Enter the amount
Exception in thread "main"
java.util.InputMismatchException
```

```
class TestAccount{
public static void main(String[] args) {
Account a= new Account();
Scanner sr = new Scanner(System.in);

System.out.println("Enter the account no");
int acc ;
acc = sr.nextInt();

System.out.println("Enter the name");
String name;
 name = sr.nextLine();

System.out.println("Enter the amount");
Float amount;
amount = sr.nextFloat();

a.insert(acc, name, amount);
a.display();
sr.close();   }}
```

Console:

```
Enter the account no
111
Enter the name
Enter the amount
Ankit
Exception in thread "main"
java.util.InputMismatchException
```

What was the problem?

- In Scanner class if we call nextLine() method after any one of the seven nextXYZ() method then the nextLine() doesn't not read values from console and cursor will not come into console it will skip that step.
  - The nextXYZ() methods are nextInt(), nextFloat(), nextByte(), nextShort(), nextDouble(), nextLong(), next().
- nextXYZ() methods ignore newline character and nextLine() only reads till first newline character.
- Solution ?

```
class TestAccount{
public static void main(String[] args) {
Account a= new Account();
Scanner sr = new Scanner(System.in);

System.out.println("Enter the account no");
int acc ;
acc = sr.nextInt();

sr.nextLine();
System.out.println("Enter the name");
String name;
name = sr.nextLine();

System.out.println("Enter the amount");
Float amount;
amount = sr.nextFloat();

a.insert(acc, name, amount);
a.display();
sr.close();   }}
```

Console:

```
Enter the account no
111
Enter the name
Ankit Tiwari
Enter the amount
5000
111 Ankit Tiwari 5000.0
```

Other differences

- BufferedReader should be used if we are working with multiple threads.
- BufferedReader has significantly larger buffer memory than Scanner.
  - The Scanner has a little buffer (1KB char buffer) as opposed to the BufferedReader (8KB byte buffer).
- BufferedReader is a bit faster as compared to scanner because scanner does parsing of input data and BufferedReader simply reads sequence of characters.