# How to get input from user in Java

## Java Scanner Class

Java **Scanner class** allows the user to take input from the console. It belongs to **java.util** package. It is used to read the input of primitive types like int, double, long, short, float, and byte. It is the easiest way to read input in Java program.

Syntax:



The above statement creates a constructor of the Scanner class having **System.in** as an argument. It means it is going to read from the standard input stream of the program. The **java.util** package should be import while using Scanner class.

**Example of integer input from user**

| 1. **Import** java.util.\*; 2. **class** UserInputDemo 3. { 4. **public** **static** **void** main(String[] args) 5. { 6. Scanner sc= **new** Scanner(System.in);    //System.in is a standard input stream 7. System.out.print("Enter first number- "); 8. **int** a= sc.nextInt(); 9. System.out.print("Enter second number- ");   **int** b= sc.nextInt();  System.out.print("Enter third number- ");  **int** c= sc.nextInt();  **int** d=a+b+c;  System.out.println("Total= " +d);  }  } |
| --- |

**Example of String Input from user:**

| 1. **import** java.util.\*; 2. **class** UserInputDemo1 3. { 4. **public** **static** **void** main(String[] args) 5. { 6. Scanner sc= **new** Scanner(System.in); //System.in is a standard input stream 7. System.out.print("Enter a string: "); 8. String str= sc.nextLine();              //reads string 9. System.out.print("You have entered: "+str);   }  } |
| --- |

**Aggregation :**( **whole** – **part** relationship)

Aggregation is a specialized form of Association in which all objects have their own lifecycle. There is a relation of ownership between 2 objects but the **part** object can exist even if we delete the **whole** object. We can think of it as “has-a” relationship. In UML, Aggregation is shown with an **open diamond**.

**Example:**





| package javaapplication5;  class Person  {  private String personName;  private int age;  Person(String personName, int age)  {  this.personName = personName;  this.age = age;  }  void printInfo()  {  System.out.println(" Person Name: "+Name+"Person Age: "+age);  }  }  class Country{  private String countryName;  private Person person;  Country(String countryName, Person person)  {  this.countryName = countryName;  this.person = person;  }  void printInfo()  {  System.out.println("Country Name: " + countryName);  person.printInfo();    }  }  public class JavaApplication5  {  public static void main(String[] args)  {  Person p;  p = new Person("Ali", 24);    Country c=new Country("Pakistan", p);  c.printInfo();  }  } |
| --- |

**Composition:**   
Composition is again a specialized form of Aggregation. It is a strong type of Aggregation. Part object does not have their lifecycle and if the whole object is destroyed then all part objects will also be destroyed along with it. In UML, **Composition** is shown with a **filled diamond**.

Example:



| package javaapplication5;  class Birthday  {  private int day;  private int month;  private int year;  Birthday(){}  Birthday(int day, int month, int year)  {  this.day = day;  this.month = month;  this.year = year;  }  void printInfo()  {  System.out.println( " Bitrhday: " + (day) + "-" + (month) + "-" + (year));  }  }  class Person  {  private String name;  private Birthday dob;  Person(String name, Birthday dob)  {  this.name = name;  this.dob = dob;  }  void printInfo()  {  System.out.println(" Name: " + name);  dob.printInfo();    }  }  public class JavaApplication5  {  public static void main(String[] args)  {  Birthday dob= new Birthday(12, 11, 2010);  Person p=new Person("Ali", dob);  p.printInfo();  }  } |
| --- |