**Functions in Java**  
A function is a block or portion of code within a larger program which consists of some declarations and executable statements, which performs a specific task and is relatively independent of the remaining code.  
The term method is used alternatively with function and means the same in Java programming language. A larger block of code is broken down into smaller and simpler modules called ‘member methods’ this is how the OOP language Java implements ‘Modularity’.  
Advantages of using Functions –  
1. To cope with the complexity of code by breaking down complex bigger code into smaller and simpler ones.  
2. Hiding details (thus implementing Abstraction)  
3. Reusability of Code  
4. Makes debugging of errors easy.  
**Anatomy of a function**  
FUNCTION NAME ( )  
{  
Body of the function…  
}  
Let us now see all the components individually –  
1. These are keywords which define the accessibility of the function, if nothing is mentioned then by default it is ‘friendly’ friendly is not a keyword. They are of 4 types  
a. Public  
b. private  
c. protected  
d. friendly this is not a keyword and hence cannot be written explicitly.  
2. These are keywords which redefines the functionality within the methods. They can be final, native, static, synchronized, transient, volatile etc…  
3. They specify what type of value a function can return, hence they can be of any primitive of reference data types like int, byte, float, char, String or even class types. In case the function doesn’t return any value the keyword ‘void’ is used.  
4. Function name – This can be any valid identifier, but it is always advisable to take logical and meaningful names, it should conventionally begin with a lowercase letter and in case of multiple words join the words and begin each word in Uppercase except for the first word.  
e.g. toUpperCase( )  
5. This is a comma separated list of variables along with associated data types. This list can also be empty which indicates the function is non–argumentative.  
e.g.  
public static void add(int a, int b)  
{  
…  
}  
**Function prototype**  
The first line of a function that consists of access specifier, modifier, return type, function name and list of parameters enclosed within a pair of parenthesis is called function prototype.  
A function prototype describes the function interface to the compiler, by giving details of number and type of arguments, return type etc…  
e.g.  
public static void compute(int a, int b)  
{  
…  
…  
}  
**Function signature**  
The function signature is a part of function prototype; it basically refers to the argument list i.e. number, order and type of arguments.  
e.g.  
public static void compute(int a, int b)  
{  
…  
…  
}  
**Understanding the keywords public & private**  
In a function the access specifier indicate the scope of visibility of that particular function, this means whether the data members and member methods are accessible only within the class, or other classes within the same package or other classes in other packages.  
Public members are accessible everywhere, within the same class as well as inside other classes that inherits the base class or by creating objects of the former class in the latter one even if they do not inherit, but private members are accessible only within the same class.  
**Understanding the keyword static**  
A static data member is also known as a class variable, such variables have only one instance inside the class and we can not create multiple instances through multiple objects.  
On the other hand if a member is non-static then such members are called instance variables, which mean we can have multiple instances of such variables through multiple objects.  
e.g.  
class Exmp1  
{  
int x; // non-static or instance variable which can have multiple instances  
static int y; // static variable or class variable which has only one instance  
public static void main( )  
{  
Exmp1 obj1 = new Exmp1( );  
Exmp1 obj2 = new Exmp1( ); // three different objects through which we can  
Exmp1 obj3 = new Exmp1( ); // have three different values of ‘x’  
obj1.x=10;  
obj2.x=20; // the same variable ‘x’ has three different values simultaneously  
obj3.x=30;  
y = 100; // the variable ‘y’ being static is accessed directly and can have only  
// one value  
obj1.y=100;  
obj2.y=200; // even if we try to access the variable ‘y’ through different objects  
obj3.y=300; // the same ‘y’ gets overwritten and finally the last value prevails  
}}  
So, we see that if a member is static in nature it will have only one copy in the physical memory of the computer, which can be accessed directly without the need of creating objects. But if it is non-static then we need to create objects to specify which instance we are referring to as it has multiple occurrences.  
**How to access a function?**  
A function can be accessed or invoked from other functions by simply writing the name of the function along with the argument list (if any) otherwise the argument list is kept empty.  
e.g.  
class ABCD  
{  
public static void main( )  
{  
int x = 10, y = 20;  
 function call⇓add (x, y);   
}  
public static void add(int x, int y)  
{  
System.out.println(x+y);  
}  
}  
**Actual & Formal Parameters / Arguments**  
The actual parameters are those that appear at the point of function invocation or function call are called Actual parameters.  
The formal parameters are those that appear at the point of function declaration or function signature are called Formal parameters.  
e.g.  
public static void main( )  
{  
int x = 10, y = 20;  
 actual parameters⇓add (x, y);   
}  
 formal parameters⇓public static void add(int x, int y)   
{  
System.out.println(x+y);  
}  
A function be of one of the four types depending on its argument list and return type –  
**1. Non-argumentative & non-return type**  
Example  
class ABCD  
{  
public static void main( )  
{  
display( );  
}  
public static void display( )  
{  
System.out.println(“Loyola School”);  
}  
}  
**2. Argumentative & non-return type**  
Example  
class ABCD  
{  
public static void main( )  
{  
int x = 10;  
display(x);  
}  
public static void display(int x)  
{  
System.out.println(“The value of x =” + x);  
}  
} // Note that a function which is void cannot return any value  
**3. Non-argumentative & return type**  
Example  
class ABCD  
{  
public static void main( )  
{  
int x = display( );  
System.out.println(“The value of x =” + x);  
}  
public static int display( )  
{  
int x = 10;  
return x;  
}  
} // Note that the return type of the function and the variable associated  
// with the keyword return must be identical, also note that the function  
// at the point of invocation gets equated with a similar datatype.  
**4. Argumentative & return type**  
Example  
class ABCD  
{  
public static void main( )  
{  
int x = 10;  
int y = display( x );  
System.out.println(“The square value of x =” + y);  
}  
public static int display(int x)  
{  
int y = x \* x;  
return y;  
}  
}  
The return statement  
A function terminates as soon as the last statement of the function is encountered or the return keyword is encountered.  
It is not necessary that return keyword should be used only in functions that has a return type, even functions that are void can use the return keyword to terminate the function and shift the program control to the calling function but make sure that the return statement does not have anything preceding the keyword ‘return’.  
e.g.  
class ABCD  
{  
public static void main( )  
{  
call( );  
}  
public static void call( )  
{  
System.out.println(“Inside the method call”);  
return ; // return statement has nothing following it  
}  
}  
**Three types of functions:**  
1. **Computational** functions – The functions that computes or calculates certain values or quantities and returns the answer to the calling method is called Computational function.  
e.g. Math.pow(a,b);  
2. **Manipulative** functions – The functions which manipulates the data and returns information to the calling function as success or failure code in the form of 0 & 1 or true & false.  
e.g. Character.isLetter( )  
3. **Procedural** Functions – The functions that perform certain tasks like reading from source files, reading from terminals, displaying data to the terminal are called procedural functions. They generally do not have any return type.  
e.g. System.out.println( );  
**Pure & Impure functions**  
A pure function is one that takes objects and/or primitive data as arguments but does not modify the objects hence the return value of such functions are either primitive data or an entirely new object created inside the function. They are also known as Accessor methods.  
An impure function on the other hand modifies the state of its objects or arguments. They are also known as Modifier methods.  
 Pure Function⇓public static int maximum(int a, int b)   
{  
int c = Math.max(a,b);  
return c;  
}  
 Impure Function⇓public static int product(int a)   
{  
a = a \* a;  
return a;  
}  
**Function Overloading**  
A function name having several definitions in the same scope that are differentiated on the basis of function signature i.e. number of arguments, order of arguments, type of arguments is said to be an overloaded function and the process of creating such functions is called Function Overloading.  
Function overloading implements **Polymorphism**.  
e.g.  
class Overload  
{  
public static void area(int s)  
{  
System.out.println(“Area of square =” + (s\*s));  
}  
public static void area(int l, int b)  
{  
System.out.println(“Area of rectangle =” + (l\*b));  
}  
public static void area(double r)  
{  
System.out.println(“Area of circle =” + (3.1415 \* r \* r));  
}

1. **Tick (√ ) the correct option.**
   1. Once a function is defined, it can be invoked repeatedly. What is this feature called?
      1. Interface b. Reusability

c. Restructuring d. All of these

**Ans.** b. Reusability

* 1. If a function does not return any value, its return type should be:
     1. int b. no-return

c. void d. empty

**Ans.** c. void

* 1. A function that computes a value and returned is called:
     1. Computational function b. Manipulative function

c. Procedural function d. None of these

**Ans.** a. Computational function

* 1. A type of parameter that are used to identify what data is to be passed to a function is called:
     1. Formal parameter b. Actual parameter

c. Both a and b d. None of these

**Ans.** a. Formal parameter

* 1. The parameter list in a function definition is called:
     1. Function prototype b. Function signature

c. Both a and b d. None of these

**Ans.** b. Function signature

* 1. The first line of the function definition is called:
     1. Function prototype b. Function signature

c. Both a and b d. None of these

**Ans.** a. Function prototype

* 1. The number of values that a function can return is:

a. 1 b. 2 c. 3 d. 4

**Ans.** a. 1

* 1. A local variable in a function has its scope:
     1. Limited within the function
     2. Can be accessed anywhere within the same class
     3. No limitation at all
     4. None of these

**Ans.** a. Limited within the function

* 1. Which among the following is a valid name for a function?
     1. function b. 2function

c. fun in action d. fun#

**Ans.** a. function

* 1. Which among the following is not a valid access specifier?
     1. public b. private

c. partially d. protected

**Ans.** c. partially

1. **Fill in the blanks.**
   1. In Java functions are known as **methods**.
   2. Methods are contained in **class**.
   3. The function name and the parameter list together is known as **function signature**.
   4. The access specifier, return type and the function signature together is known as **prototype**.
   5. The arguments of the function given in function prototype are called **formal arguments**.
   6. The arguments of the function given in statement that calls the function are called **actual arguments**.
   7. If a function does not return any value, the returning type in the function prototype will be **void**.
   8. When a function is called by **value**, the values of actual parameters are copied into separate memory locations as allocated by the formal parameters.
   9. Impure functions uses call by **reference**.
   10. One of the practical implementation of polymorphism is **overloading**.

# SECTION A

1. **Answer the following questions.**

### What is a method?

**Ans.** A Java method is a collection of statements that are grouped together to perform an operation.

### Write two advantages of using functions in a program.

**Ans.** Reduces complexity and Reusability.

### Explain the function of the ‘return’ statement.

**Ans.** The ‘return’ statement is used to return a value from a method, before exiting from the function.

### If a function contains several return statements, how many of them will be executed?

**Ans.** The first one.

### Name the keyword that causes the control to transfer back to the method call. Ans. return

* 1. **What is the role of the keyword ‘void’ in declaring functions?**

**Ans.** The void keyword ensures that a method do not return any value.

### Classify functions depending upon the value it returns.

**Ans.** Computational function, Procedural functions and Manipulative functions

### Differentiate between Formal Parameter and Actual Parameter.

**Ans.** Difference:

|  |  |
| --- | --- |
| **Formal Parameters** | **Actual Parameters** |
| Parameter list in the function prototype. | Parameter list in the function invocation. |
| They are only variables. | They may be both variables or constants. |

### State the difference between function prototype and function signature.

**Ans.** Function prototype is the first line of the function definition that consist of return type function name and the parameter list. Function signature on the other hand only specifies the parameter list.

### How are functions called? How does it return a value?

**Ans.** Functions are called using the name of the function, they may be called using either:

* + - Call by Value • Call by reference

The return statement is used to return a value from a function.

### State the difference between Call by Value and Call by Reference.

**Ans.**

|  |  |
| --- | --- |
| **Difference between Call by Value and Call by Reference** | |
| **Call by Value** | **Call by Reference** |
| Using this technique a copy of the values in the actual parameter is made in the formal parameters. | Using this technique a reference to the values in the actual parameter is made by the formal parameters. |

|  |  |
| --- | --- |
| Any changes made to the formal parameters is not reflected back in the actual parameters. | Any changes made to the formal parameters is reflected back in the actual parameters. |
| Usually primitive data type is used during call by value. | Usually arrays and objects are used during call by reference. |

### How are the following passed?

1. **Primitive types (ii) Reference types**

**Ans.** (i) Call by value

1. Call by reference

### Differentiate between pure and impure function.

**Ans.**

|  |  |
| --- | --- |
| **Difference between Pure and Impure Functions** | |
| **Pure Functions** | **Impure Functions** |
| Pure functions are such functions which do not change the state of an object. | Impure functions are such functions which changes the state of an object. |
| It doesn’t have any side effects as the state of the object is not changed, rather only accessed. | It does have side effects as the state of the object is changed, and therefore one should be careful using it. |

### Explain function overloading with an example.

**Ans.**

Example of function overloading: class Overload

{

static void num\_calc(int num,char ch)

{

if(ch==’s’) System.out.println(“Square:”+(num\*num));

else

System.out.println(“Square:”+(num\*num\*num));

}

static void num\_calc(int a,int b,char ch)

{

if(ch==’p’) System.out.println(“Product:”+(a\*b));

else

System.out.println(“Sum:”+(a+b));

}

}

### Which OOP principle implements function overloading?

**Ans.** Polymorphism

### When there are multiple definitions with the same function name, what makes them different from each other?

**Ans.** Function signature

### What are the different access specifiers available in Java?

**Ans.** default, public, private and protected

### What is the use of main() method?

**Ans.** The main() method, in general, is from where a program execution begins in Core Java.

### How are static methods of one class called by methods in other classes?

**Ans.** The function invocation should be preceded with the name of the class followed by the dot

operator.

1. **Write function prototypes for the following:**

### Private access method sum which accepts three int type variables as parameters and return a

**float type.**

**Ans.** private float sum(int a,int b,int c)

### Default access method “length” which accepts a String type variable as parameter and returns an int type.

**Ans.** int length(String s)

### Public access method “increment” which accepts an object of Myclass type as parameter and does not return any value.

**Ans.** public void increment(Myclass ob)

### Protected access method largest which accepts a float type, int type and double type data as

**parameters and have a byte type as return type.**

**Ans.** protected byte largest(float a, int b, double c)

### Public access static method calculate which accepts a byte and int type data type as parameters and return float type.

**Ans.** public static float calculate(byte b, int d)

### Write the function prototype for the function “sum” that takes an integer variable (x) as its argument and returns a value of float data type.

**Ans.** float sum(int x)

### Write the prototype of a function which takes in 2 integers and 1 String arguments and returns

**a value which is either ‘true’ or false’. Ans.**

boolean function(int a,int b,String c)

---------------------------------------------------------------------------------------------------------------------------------------

1.What is Function? Why do we use functions while programs handling?

Ans: A named unit of a group of programs statements. This unit can be invoked from other parts of the program.

1. Define Function prototype?

Ans: The function prototype is the first line of the function definition that tells the program about the type of the value returned by the function and the number and types of arguments.

1. What is the use of void before function name? [2007]

Ans: void data type specifies an empty set of values and it is used as the return type for functions that do not return a value. Thus a function that does not return a value is declared as follows. void <functions name> (parameter list)

1. Explain Functions/Methods Definitions with syntax?

Ans: A function must be defined before it is used anywhere in the program.

[access specifier][modifier]return-type function-name (parameter list)

{

body of the function

}

[access specifier] can be either Public, Protected or Private. [modifier] can be one of final, native, synchronize, transient, volatile. return-type specifies the type of value that the return statement of the function returns. It may be any valid Java data type. parameter list is comma separated list of variables of a function.

1. Why main() function so special?

Ans: The main() function is invoked in the system by default. hence as soon as the command for execution of the program is used, control directly reaches the main() function.

1. Explain the function prototype and the signature?

Ans: The function prototype is the first line of the function definitions, that tells the program about the type of the value returned by the function and the number and type of the arguments. Function signature basically refers to the number and types of the arguments, it is the part of the prototype.

1. Explain the function of a return statement? [2006]

Ans: The return statement is useful in two ways. First an immediately exit from the function is caused as soon as a return statement is encountered and the control back to the main caller. Second use of return statement is that it is used a value to the calling code.

1. Write advantages of using functions in programs

Ans: (i) functions lessen the complexity of programs (ii) functions hide the implementation details (iii) functions enhance reusability of code

1. Difference between Actual argument and Formal argument? [2007,2008]

Ans: The parameter that appears in function call statement are called actual argument and The parameter that appears in function definition are called formal parameter.

1. What are static members?

Ans: The members that are declared static is called static members. These members are associated with the class itself rather then individual objects, the static members and static methods are often referred to as class variables and methods.

1. What is the use of static in main() methods? [2007]

Ans: (i) They can only call other static methods. (ii) They can only access static data. (iii) They can not refer to this or super in any way.

1. What is call by value? [2005]

Ans: (i) In call by value, the called functions creates its own work copy for the passed parameters and copies the passed values in it. Any changes that take place remain in the work copy and the original data remains intact.

1. Explain the term “passed by reference”? [2007]

Ans: In passed by reference, the called function receives the reference to the passed parameters and through this reference, it access the original data. Any changes that take place are reflected in the original data.

1. Differentiate between call by value and call by reference?

Ans: In call by value, the called functions creates its own work copy for the passed parameters and copies the passed values in it. Any changes that take place remain in the work copy and the original data remains intact. In call by reference, the called function receives the reference to the passed parameters and through this reference, it access the original data. Any changes that take place are reflected in the original data.

1. Define an impure functions? [2006]

Ans: Impure Function change the state of the object arguments they have received and then return. The following functions is the example of an impure function:

public static void increment(Time obj, double secs)

{

time.seconds+=secs;

return(Time);

}

1. What is the difference between pure and impure functions? [2009]

Ans: Pure Function: These functions takes objects as an arguments but does not modify the state of the objects. The result of the pure function is the return value. Impure Function: These functions change the state of the object arguments they have received.

1. How are following passed in Java? [2005]

(i) primitive types    (ii) reference types

Ans: (i) By value,     (ii) By reference.

1. What does function overloading mean? What is its significance?

Ans: A Function name having several definitions in the same scope that are differentiable by the number or type of their arguments, is said to be an overloaded function. Function overloading not only implements polymorphism but also reduce the number of comparisons in a program and there by makes the programs run faster.

1. Illustrate the concept of function overloading with the help of an example. [2006]

Ans:- A function name having several definitions that are differentiable by the numbers or types of their arguments is known as function overloading. For example following code overloads a function area to computer areas of circle rectangle and triangle.

float area (float radius)            //circle

{

return (3.14 \* radius \* radius);

}

float area (float length, float breadth)  //rectangle

{

return (length\*breadth);

}

float area (float side1, float  side2, float side3) //area of triangle

{

float s = (side1 + side2 + side3)/2;

float ar = Math.sqrt(s \* (s- side1)\*(s-side2) \*(s-side3));

return (ar);

}

1. What is this keyword? What is its significance? [2009]

Ans: The this keyword is used to refer to currently calling objects. The member functions of every objects have access to a sort of magic keyword name this, which points to the object itself. Thus any member function can find out the address of the object of which it is a member. The this keyword represents an object that invokes a member function. it stores the address of the object that invoking a member function and it is an implicit argument to the member function being invoked. The this keyword is useful in returning the object of which the function is a member.

1. What do you mean by recursive function?

Ans: When a method is called inside its own definition the process is known as functions recursion and this function called recursive function.

1. What is the difference between Methods and Functions?

Ans: The major difference between methods and functions is that methods called by the reference variables called objects where as the functions do not having any reference variables.