Assignment 7 Solution

Theory Questions with Answers:

1. Differentiate between:

a. Call by Value vs Call by Reference -

Ans:

Call By Value	***Call By Reference***
• In Call by Value, the duplicate copies of the variable are created. Therefore, whatever modifications done inside the function gets reflected in the duplicate copies, while original copies remain unaffected.	• In Call by Reference, no duplicate copies of the variable are created. Therefore, whatever modifications done inside the function gets reflected in the original copies of a variable.
• In this method, primitive types are passed.	• In this method, non-primitive types are always references.

b. Function and Constructors-

Ans:

Function	***Constructors***
• Function is a group of statements that can be called at any point in the program using its name to perform a specific task.	• Constructor is a block of code that initializes a newly created object.
• Function should have a different name than class name.	• Constructor has the same name as class name.

c. Actual Parameters vs Formal Parameters

Ans:

Actual Parameters	***Formal Parameters***
• The parameters which appears in function calling statement are called actual parameters.	• The parameters which appears in function definition/prototype are called formal parameters

• Example: -
`public static void main(String Args[])`
`{`
`    Test ob= new Test();`
`   ob.large(5,10);`
`   int x=5,y=10;`
`   ob.large(x,y);`
`}`
	• Example:-
	`class test{`
	`   void large(int a, int b)`
	`   {`
	`      // Body of the function`
	`   }`

d. Private vs Public

Ans:

Private	***Public***
• It is the least restricted access specifier.	• It is the most restricted access specifier.
• Members declared under this section are accessible in all parts of Java program.	• Members declared under this section are accessible only in its own class.

e. Instance Variables vs Class Variables

Ans:

Class Variables	***Instance Variables***
• A variable which maintains a single copy for the whole class which is shared by all the objects of that class is called Class Variables.	• A variable which maintains a separate copy for each and every individual object is called instance variables.
• The static variable gets memory only once in the class area at the time of class loading.	• The non-static variable gets memory every time an object is created.
• It makes your program **memory efficient** (i.e., it saves memory).	• It wastes lot of memory.

• Example:- class demo{
 static int a; //*Class variable*
 }
	• Example:- class demo{
	 int a; //*Instance variable*
	 }

f. Instance Methods Vs Class Methods

Ans:

Class Methods	***Instance Methods***
• The methods which are executed through the class name are called Class Methods.	• The methods which are executed through the objects of the class are called instance variables.
• Example: - Math.pow;// here **Math** is the class name and **pow()** is the function in it.	• Example:- sc.nextInt(); // Here we are using the function **nextInt()** with the help of its object **sc.**
• A static method can only access static data member and can change the value of it. It cannot access non-static variable.	• It can use both instance and class variables.

2. WAP to create function char isPalin(int n) & float isAuto(int n) & print the PalinAuto numbers between 1 to 100.

```
class palAuto {
  char isPalin(int n) {
    int d,r=0,t;
    t=n;
    while(t>0) {
      d= t%10;
      r= r*10+d;
      n=n/10;
    }
    if(r==n)
      return 'y';
    else
      return 'n';
  }
  float isAuto(int n)  {
   int c=0, sqr = n*n;
   int temp=n;
   while(temp>0) {
     c++;
     temp/=10;
    }
double lastSquareDigits = sqr%(Math.pow(10,c));
   if(num==lastSquareDigits)
          return 1.0;
    else
       return 0.0;
   }
   public static void main(String Args[])  {
    palAuto ob = new palAuto();
    char x;
    float a;
    for(int i= 1; i<=100; i++)  {
       x= ob.isPalin(i);
       a= ob.isAuto(i);
       if(x=='y' && a==1.0)
         System.out.println(i);
    }
   }
}
```

3. WAP to overload functions:

- double series(int n): to print the sum of series:
  S= 1 + 1/2 + 1/3 + 1/4 + ... n terms
- double series(double a, double b): print the sum of series-
  S = $\frac{1}{a^2}+\frac{4}{a^5}+\frac{7}{a^8}+\cdots n\ terms$

Ans:

```
class overload6{
   public static void main(){
      overload6 ob = new overload6();
      ob.series(10);
      ob.series(2.0,6.0);
   }
   double series(int n){
     double s=0.0;
     for(int i=1;i<=n;i++){
        s=s+(1/i);
     }
     return s;

   }
   double series(double a,double n){
      double s= 0.0;
      for(int i=1;i<=n;i+=3){
         s=s+(i/Math.pow(a,i+1));
      }
      return s;
   }
}
```

4. **What is Function Overloading. Explain with example**
   Ans: When two or more function have same name but different signatures, they are called overloaded functions and the process of creating and calling their overloaded methods is called function overloading. It implements polymorphism. There is no role of return type in function signature and also in function overloading.
   Example:

```
class overload{
    public static void main(){
        overload ob = new overload();
        ob.compare(3,7);
        ob.compare('#','$');
    }
    void compare(int a, int b){
       if(a>b)
         System.out.println(a);
       else
        System.out.println(b);
    }
    void compare(char a, char b){
        if(a>b)
         System.out.println(a);
       else
        System.out.println(b);
    }
}
```

5. **Predict the output:**

```
class T {
   int t = 20;
 public static void main( ){
   T t1 = new T();
   System.out.println(t1.t);
}
}
```

Ans: The value of t will be printed i.e., 20.