Oracle Java 8 Certification
OCA (1Z0-808)
Course
Question Bank

Java SE 8 Programmer I(1Z0-808) Question Bank With Answers

Topic-1: Java Features

Q1. Which of the following are true?

A) Java is platform dependent but JVM is platform independent
B) Java is platform independent but JVM is platform dependent
C) Java Byte code is platform dependent but JVM is platform independent
D) Java Byte code is platform independent but JVM is platform dependent

Answer: B and D

Explanation:
Java follows Write Once and Run anywhere policy (WORA). i.e Once we write a java program, we can run on any platform without making any changes.

First Java Source File will be compiled into ByteCode. Bytecode is an intermediate and machine independent code. JVM will interpret byte code into the corresponding machine dependent code and executes that machine code.

Hence Java is Platform independent
Bytecode is Platform Independent
But JVM is Platform Dependent

Q2. Which Statement is true about Java Byte code?

A) It can run on any platform
B) It can run on any platform only if it was compiled for that platform
C) It can run on any platform that has the Java Runtime Environment (JRE)
D) It can run on any platform that has a Java Compiler
E) It can run on any platform only if that platform has both JRE and Java Compiler

Answer: C

Explanation:
Java follows Write Once and Run anywhere policy (WORA). i.e Once we write a java program, we can run on any platform without making any changes.

First Java Source File will be compiled into ByteCode. Bytecode is an intermediate and machine independent code. JVM will interpret byte code into the corresponding machine dependent code and executes that machine code.

Hence Java is Platform independent
Bytecode is Platform Independent
But JVM is Platform Dependent

Topic-2: Data Types and Literals

Q1. Which of the following conversions will be performed automatically in Java ?

A) int to byte
B) byte to int
C) float to double
D) double to float
E) None of the above

Answer: B, C

Explanation:

The following are valid implicit conversions in java

byte-->short-->int-->long-->float-->double

char-->int-->long-->float-->double

Except these in the remaining cases explicit type casting is required.

Q2. In which of the following cases explicit Type casting is required ?

A) int to byte
B) byte to int
C) float to double
D) double to float
E) None of the above

Answer: A, D

Explanation:

The following are valid implicit conversions in java

byte-->short-->int-->long-->float-->double

char-->int-->long-->float-->double

Except these in the remaining cases explicit type casting is required.

Q3. Consider the code

```
int i =100;
float f = 100.100f;
double d = 123;
```

Which of the following assignments won't compile?
A) i=f;
B) f=i;
C) d=f;
D) f=d;
E) d=i;
F) i=d;

Answer: A,D,F

Explanation:

The following are valid implicit conversions in java

byte-->short-->int-->long-->float-->double

char-->int-->long-->float-->double

Except these in the remaining cases explicit type casting is required.

Q4. In which of the following cases we will get Compile time error?

A) float f =100F;
B) float f =(float)1_11.00;
C) float f =100;
D) double d = 203.22;
float f = d;
E) int i =100;
float f=(float)i;

Answer: D

Explanation:
We cannot assign double value to the float. Explicit type casting must be required.

The following are valid implicit conversions in java

byte-->short-->int-->long-->float-->double
char-->int-->long-->float-->double

Except these in the remaining cases explicit type casting is required.

Q5. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int a=10;
6)        float b=10.25f;
7)        double c=100;
8)        a = b;
9)        b = a;
10)       c = b;
11)       c = a;
12)    }
13) }
```

Which change will enable the code fragment to compile successfully?

A) Replace a = b; with a=(int)b;
B) Replace b = a; with b=(float)a;
C) Replace c = b; with c=(double)b;
D) Replace c = a; with c=(double)a;

Answer: A

Explanation:

The following are valid implicit conversions in java

byte-->short-->int-->long-->float-->double

char-->int-->long-->float-->double

Except these in the remaining cases explicit type casting is required.

Topic-3: Arrays

Q1. Consider the folliwng code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String[] courses={"Java","Python","Testing","SAP"};
6)        System.out.println(courses.length);
7)        System.out.println(courses[1].length());
8)
9)     }
10) }
```

What is the output?

A) 4
6
B) 4
4
C) 6
4
D) 6
6
Answer: A

Explanation:
length variable applicable for arrays where as length() method applicable for String objects.
length variable represents the number of elements present in the array.
length() method returns the number of characters present in the String.

Q2. Given the following code

```
1)  int[] x= {10,20,30,40,50};
2)  x[2]=x[4];
3)  x[4]=60;
```

After executing this code Array elements are

A) 10,20,30,40,50
B) 10,20,50,40,50
C) 10,20,50,40,60
D) 10,20,30,50,60

Answer: C

Q3. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int[] n1= new int[3];
6)        int[] n2={10,20,30,40,50};
7)        n1=n2;
8)        for(int x : n1)
9)        {
10)          System.out.print(x+":");
11)       }
12)
13)    }
14) }
```

What is the output?

A) 10:20:30:40:50:
B) 10:20:30:
C) Compilation fails
D) ArrayIndexOutOfBoundsException at runtime

Answer: A

Q4. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        /* Line-1: insert code here */
6)        x[0]=10;
```

```
7)        x[1]=20;
8)        System.out.println(x[0]+":"+x[1]);
9)     }
10) }
```

Which code fragment required to insert at Line-1 to produce output 10:20

A) int[] x = new int[2];
B) int[] x;
x = int[2];
C) int x = new int[2];
D) int x[2];

Answer: A

Q5. Given Student class as

```
1)  class Student
2)  {
3)     int rollno;
4)     String name;
5)     public Student(int rollno,String name)
6)     {
7)        this.rollno=rollno;
8)        this.name=name;
9)     }
10) }
11) public class Test
12) {
13)    public static void main(String[] args)
14)    {
15)       Student[] students ={
16)          new Student(101,"Durga"),
17)          new Student(102,"Ravi"),
18)          new Student(103,"Shiva"),
19)          new Student(104,"Pavan")
20)       };
21)       System.out.println(students);
22)       System.out.println(students[2]);
23)       System.out.println(students[2].rollno);
24)    }
25) }
```

What is the output?

A) students
Shiva
103
B) [LStudent;@61baa894
Shiva
103
C) [LStudent;@61baa894
Student@66133adc
103
D) [LStudent;@61baa894
Pavan
103

Answer: C

Explanation:

Every Array is an object and for every array type corresponding classes are available internally.
Whenever we are trying to print array reference internally toString() method will be called,which returns the string in the following format:

classname@hexadecimal_String_of_hashcode

Q6. The following code creates the state of a 2D array in the form of grid:

```
char[][] grid= new char[3][3];
grid[0][0]='Y';
grid[0][1]='Y';
grid[1][1]='X';
grid[1][2]='Y';
grid[2][1]='X';
grid[2][2]='X';
// Line-1
```

Which line inserted at Line-1 so that grid contains 3 consecutive X's?

A) grid[3][1]='X';
B) grid[0][2]='X';
C) grid[1][3]='X';
D) grid[2][0]='X';
E) grid[1][2]='X';

Answer: D

Q7. Consider the code

```
1)  class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String[] s = new String[2];
6)        int i =0;
7)        for(String s1 : s)
8)        {
9)           s[i].concat("Element "+i);
10)          i++;
11)       }
12)       for(i=0;i<s.length;i++)
13)       {
14)          System.out.println(s[i]);
15)       }
16)    }
17) }
```

What is the result?
A) Element 0
Element 1

B) null Element 0
null Element 1

C) null
null

D) NullPointerException

Answer: D

Explanation: Once we creates an array,every array element is intialized with default values. For the String it is null.
On null,if we are trying to perform any operation then we will get NullPoinerException

Q8. Consider the code

```
1)  class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int[][] n = new int[1][3];
6)        for(int i =0; i< n.length; i++)
7)        {
8)           for (int j=0;j>n[i].length ;j++ )
9)           {
10)             n[i][j]=10;
11)          }
12)       }
13)    }
14) }
```

Which option represents the state of the array after successful completion of outer for loop?

A)
n[0][0]=0
n[0][1]=0
n[0][2]=0

B)
n[0][0]=10
n[1][0]=10
n[2][0]=10

C)
n[0][0]=10
n[0][1]=0
n[0][2]=0

D)
n[0][0]=10
n[0][1]=10
n[0][2]=10
n[1][0]=0
n[1][1]=0
n[1][2]=0
n[1][3]=0

Answer: A

Explanation: Once we creates an array,every array element is intialized with default values. For the int type it is 0.

Q9. Consider the code

```
1)  class Student
2)  {
3)     String name;
4)     public Student(String name)
5)     {
6)        this.name=name;
7)     }
8)  }
9)  public class Test
10) {
11)    public static void main(String[] args)
12)    {
13)       Student[] students = new Student[3];
14)       students[1]= new Student("Durga");
15)       students[2]= new Student("Ravi");
16)       for(Student s : students)
17)       {
18)          System.out.println(s.name);
19)       }
20)    }
21) }
```

What is the result?
A) Durga
Ravi

B) null
Durga
Ravi

C) Compilation Fails
D) ArrayIndexOutOFBoundsException
E) NullPointerException

Answer: E

Explanation: Once we creates an array,every array element is intialized with default values. For the String it is null.

In the above example we didn't initialize first element and hence it is null.
On null, if we are trying to perform any operation then we will get NullPoinerException

Topic-4: Types of Variables

Q1. Consider the following code

```
1)  public class  Test
2)  {
3)     String myStr="7777";
4)     public void doStuff(String s)
5)     {
6)        int myNum=0;
7)        try
8)        {
9)           String myStr=s;
10)          myNum=Integer.parseInt(myStr);
11)       }
12)       catch(NumberFormatException e)
13)       {
14)          System.err.println("Error");
15)       }
16)       System.out.println("myStr: "+myStr+" ,myNum: "+myNum);
17)   }
18)   public static void main(String[] args)
19)   {
20)      Test t = new Test();
21)      t.doStuff("9999");
22)   }
23) }
```

What is the result?

A) myStr: 9999 ,myNum: 9999
B) myStr: 7777 ,myNum: 7777
C) myStr: 7777 ,myNum: 9999
D) Compilation Fails

Answer: C

Explanation:In the above example, the variable myStr,which is declared inside try block is not available outside of try block.Hence while printing, instance variable myStr will be considered.

Q2. Consider the following code

```
1)  public class  Test
2)  {
3)     public void doStuff(String s)
4)     {
5)        int myNum=0;
6)        try
7)        {
8)           String myStr=s;
9)           myNum=Integer.parseInt(myStr);
10)       }
11)       catch(NumberFormatException e)
12)       {
13)          System.err.println("Error");
14)       }
15)       System.out.println("myStr: "+myStr+" ,myNum: "+myNum);
16)   }
17)   public static void main(String[] args)
18)   {
19)      Test t = new Test();
20)      t.doStuff("9999");
21)   }
22) }
```

What is the result?

A) myStr: 9999 ,myNum: 0
B) myStr: 0 ,myNum: 0
C) myStr: 9999 ,myNum: 9999
D) Compilation Fails

Answer: D

Explanation: myStr is local variable of try block and we cannot access outside of try block.

Q3. Consider the code

```
1)  class Test
2)  {
3)     int x =10;
4)     static int y = 20;
5)     public static void main(String[] args)
```


```
6)    {
7)       Test t1 =new Test();
8)       Test t2 =new Test();
9)       t1.x=100;
10)      t1.y=200;
11)      t2.x=300;
12)      t2.y=400;
13)      System.out.println(t1.x+".."+t1.y+".."+t2.x+".."+t2.y);
14)   }
15) }
```

What is the result?

A) 100..400..300..400
B) 100..400..100..400
C) 200..400..300..400
D) 100..200..300..400

Answer: A

Q4. Consider the following code

```
1)  public class Test
2)  {
3)     static int x;
4)     int y;
5)     public static void main(String[] args)
6)     {
7)        Test t1 = new Test();
8)        Test t2 = new Test();
9)        t1.x=3;
10)       t1.y=4;
11)       t2.x=5;
12)       t2.y=6;
13)       System.out.println(t1.x+":"+t1.y+":"+t2.x+":"+t2.y);
14)    }
15) }
```

What is the result?
A) 3:4:5:6
B) 3:4:3:6
C) 5:4:5:6
D) 3:6:4:6

Answer: C

Explanation:

In the case of instance variables, for every object a seperate copy will be created. Hence by using one object reference if we are trying to perform any change,those changes won't be reflected to the remaining objects.

But in the case of static variables only one copy will be created at class level. By using any object reference if we are trying to perform changes then those changes will be reflected to all the remaining objects also.

Q5. Consider the code

```
1)  public class Test
2)  {
3)     static int count=0;
4)     int i =0;
5)     public void modify()
6)     {
7)        while(i<5)
8)        {
9)           i++;
10)          count++;
11)       }
12)    }
13)    public static void main(String[] args)
14)    {
15)       Test t1 = new Test();
16)       Test t2 = new Test();
17)       t1.modify();
18)       t2.modify();
19)       System.out.println(t1.count+".."+t2.count);
20)    }
21) }
```

What is the result?
A) 10..10
B) 5..5
C) 5..10
D) Compilation Fails

Answer: A

Explanation:

In the case of instance variables, for every object a seperate copy will be created. Hence by using one object reference if we are trying to perform any change,those changes won't be reflected to the remaining objects.

But in the case of static variables only one copy will be created at class level. By using any object reference if we are trying to perform changes then those changes will be reflected to all the remaining objects also.

Q6. Consider the code

```
1)  class Test
2)  {
3)     int count;
4)     public static void display()
5)     {
6)        count++;//Line-1
7)        System.out.println("Welcome Visit Count:"+count);//Line-2
8)     }
9)     public static void main(String[] args)
10)    {
11)       Test.display();//Line-3
12)       Test.display();//Line-4
13)    }
14) }
```

What is the result?

A) Welcome Visit Count: 1
** Welcome Visit Count: 2**
A) Welcome Visit Count: 1
** Welcome Visit Count: 1**
C) Compilation Fails at Line-1 and Line-2
D) Compilation Fails at Line-3 and Line-4

Answer: C
Explanation:
Non-static variables cannot be accessed directly from the static area. Hence we are getting compile time error at Line-1 and Line-2

Q7. Consider the code

```
1)  public class Test
2)  {
3)     public static int x=100;
4)     public int y = 200;
5)     public String toString()
6)     {
7)        return y+":"+x;
8)     }
9)     public static void main(String[] args)
10)    {
11)       Test t1 = new Test();
12)       t1.y=300;
13)       System.out.println(t1);
14)       Test t2 = new Test();
15)       t2.x=300;
16)       System.out.println(t2);
17)    }
18)
19) }
```

What is the result?

A) 300:100
200:300

B) 200:300
200:300

C) 300:0
0:300

D) 300:300
200:300

Answer: A

Explanation:

In the case of instance variables, for every object a seperate copy will be created. Hence by using one object reference if we are trying to perform any change,those changes won't be reflected to the remaining objects.

But in the case of static variables only one copy will be created at class level. By using any object reference if we are trying to perform changes then those changes will be reflected to all the remaining objects also.

Q8. Consider the code

```
1)  public class  Test
2)  {
3)     static double area;
4)     int b=30,h=40;
5)     public static void main(String[] args)
6)     {
7)        double p,b,h;// Line-1
8)        if(area ==0)
9)        {
10)          b=3;
11)          h=4;
12)          p=0.5;
13)       }
14)       area=p*b*h;// Line-2
15)       System.out.println(area);
16)    }
17) }
```

What is the result?
A) 6.0
B) 3.0
C) Compilation fails at Line-1
D) Compilation fails at Line-2

Answer: D
Explanation:

For instance and static variables JVM will provide default values and hence we are not required to perform initialization explicitly.
But for local variables JVM won't provide default values and compulsory we should provide intialization explicitly before using that variable,otherwise we will get compile time error. For local variables it is not recommended to initialize inside logical blocks like if block, because there is no guarantee for the execution of logical block always.

Q9. Consider the code

```
1)  class Demo
2)  {
3)     int ns;
4)     static int s;
5)     Demo(int ns)
```

```
6)    {
7)       if(s<ns)
8)       {
9)          s=ns;
10)         this.ns=ns;
11)      }
12)   }
13)   void display()
14)   {
15)      System.out.println("ns = "+ns+"  s = "+s);
16)   }
17)}
18)public class Test
19){
20)   public static void main(String[] args)
21)   {
22)      Demo d1= new Demo(50);
23)      Demo d2= new Demo(125);
24)      Demo d3= new Demo(100);
25)      d1.display();
26)      d2.display();
27)      d3.display();
28)   }
29) }
```

What is the result?

A)
ns = 50 s = 125
ns = 125 s = 125
ns = 0 s = 125

B)
ns = 50 s = 125
ns = 125 s = 125
ns = 100 s = 125

C)
ns = 50 s = 50
ns = 125 s = 125
ns = 0 s = 0

D)
ns = 50 s = 125
ns = 125 s = 125
ns = 100 s = 100

Answer: A
Explanation:

In the case of instance variables, for every object a seperate copy will be created. Hence by using one object reference if we are trying to perform any change,those changes won't be reflected to the remaining objects.

But in the case of static variables only one copy will be created at class level. By using any object reference if we are trying to perform changes then those changes will be reflected to all the remaining objects also.

Q10.
Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int x;
6)        if (args.length>0)
7)        {
8)           x=10;
9)        }
10)       System.out.println(x);
11)    }
12) }
```

If we run the code by "java Test 1 2 3 4". What is the result?

A) Compilation fails.
B) An exception is thrown at runtime.
C) 10
D) 10 10 10 10

Answer: A
Explanation:
If we are not passing any command line argument then x won't be initialized. Hence there is guarantee for initialization of x always.

Before accessing a local variable compulsory we should perform initialization. Hence we will get Compile Time Error saying: variable x might not have been initialized

Q11. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        try
6)        {
7)           int n=10;
8)           int d=0;
9)           int ans=n/d;
10)       }
11)       catch (ArithmeticException e)
12)       {
13)          ans=0;//Line-1
14)       }
15)       catch(Exception e)
16)       {
17)          System.out.println("Invalid Calculation");
18)       }
19)       System.out.println("Answer="+ans);//Line-2
20)   }
21) }
```

What is the result?

A) Answer=0
B) Invalid Calculation
C) Compiation Fails at Line-1
D) Compiation Fails at Line-2
E) Compiation Fails at Line-1 and Line-2

Answer: E

Explanation:

The local variables which are declared inside a block are accessible within the block only. outiside of that block we cannot access.

In the above code ans is the local variable declared inside try block and cannot be accessed outside of try block. Hence we are getting compile time errors at Line-1 and Line-2

Q12. Consider the code

```
1)  public class Test
2)  {
3)     char c;
4)     boolean b;
5)     float f;
6)     public void print()
7)     {
8)        System.out.println("c = "+c);
9)        System.out.println("b = "+b);
10)       System.out.println("f = "+f);
11)    }
12)    public static void main(String[] args)
13)    {
14)       Test t = new Test();
15)       t.print();
16)    }
17) }
```

What is the result?

A)
c =
b = false
f = 0.0

B)
c = null
b = false
f = 0.0

C)
c = 0
b = false
f = 0.0

D)
c =
b = true
f = 0.0

Answer: A

Explanation:

For instance and static variables JVM will provide default values and we are not required to perform initialization explicitly.

The default value for char is space character
The default value for boolean is false
The default value for float is 0.0

Parameter Passing:

Q13. Consider the code

```
1)  public class Test
2)  {
3)     public void m1(int i, int j)
4)     {
5)        i=i+10;
6)        j=j+10;
7)        System.out.println("Inside Method:"+i+".."+j);
8)     }
9)     public static void main(String[] args)
10)    {
11)       int x=100;
12)       int y =200;
13)       Test t = new Test();
14)       t.m1(x,y);
15)       System.out.println("After Completing Method:"+x+".."+y);
16)    }
17) }
```

What is the result?

A.
Inside Method:110..210
After Completing Method:100..200

B.
Inside Method:100..210
After Completing Method:100..200

C.
Inside Method:110..200
After Completing Method:100..200

D.
Inside Method:100..200
After Completing Method:110..210

Answer: A

Explanation:

If we are passing primitive values to a method as argument and if we are changing any changes to the variable inside method then those changes won't be reflected to the caller because a seperate copy will send to the method.

But if we pass object reference as argument to a method and if we are performing any changes to that object in the method body,those changes will be reflected to the caller also.

Q14. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int x=200;
6)        System.out.print(m1(x));
7)        System.out.print(" "+x);
8)     }
9)     public static int m1(int x)
10)    {
11)       x= x*2;
```


```
12)       return x;
13)    }
14) }
```

What is the result?

A) 400 200
B) 200 200
C) 400 400
D) Compilation Fails

Answer: A

Explanation:

If we are passing primitive values to a method as argument and if we are changing any changes to the variable inside method then those changes won't be reflected to the caller because a seperate copy will send to the method.

But if we pass object reference as argument to a method and if we are performing any changes to that object in the method body,those changes will be reflected to the caller also.

Q15. Consider the following code

```
1)  public class Test
2)  {
3)     int i,j;
4)     public Test(int i,int j)
5)     {
6)        initialize(i,j);
7)     }
8)     public void initialize(int i,int j)
9)     {
10)       this.i = i*i;
11)       this.j=j*j;
12)    }
13)    public static void main(String[] args)
14)    {
15)       int i =3, j= 5;
16)       Test t = new Test(i,j);
17)       System.out.println(i+"..."+j);
18)    }
```


```
19) }
```

What is the result?

A) 9...25
B) 0...0
C) 3...5
D) Compilation Fails

Answer: C

Explanation:

If we are passing primitive values to a method as argument and if we are changing any changes to the variable inside method then those changes won't be reflected to the caller because a seperate copy will send to the method.

But if we pass object reference as argument to a method and if we are performing any changes to that object in the method body,those changes will be reflected to the caller also.

Q16. Consider the code

```
1)  class Demo
2)  {
3)     int x;
4)     int y;
5)  }
6)  public class Test
7)  {
8)     public void m1(Demo d)
9)     {
10)       d.x=888;
11)       d.y=999;
12)    }
13)    public static void main(String[] args)
14)    {
15)       Demo d1 = new Demo();
16)       d1.x=10;
17)       d1.y=20;
18)       Test t = new Test();
19)       t.m1(d1);
20)       System.out.println(d1.x+"..."+d1.y);
```

```
21)    }
22)  }
```

What is the result?

A. 888...999
B. 888...20
C. 10...999
D. 10...20

Answer: A

Explanation:

In the above example we are passing Demo object reference as argument to method m1().Inside method m1(),we are changing the state of the object. These changes will be reflected to the caller.

Q17. Consider the code

```
1)  class Product
2)  {
3)     double price;
4)  }
5)  public class Test
6)  {
7)     public void updatePrice(Product p,double price)
8)     {
9)        price=price*2;
10)       p.price=p.price+price;
11)    }
12)    public static void main(String[] args)
13)    {
14)       Product prt = new Product();
15)       prt.price=200;
16)       double newPrice=100;
17)
18)       Test t = new Test();
19)       t.updatePrice(prt,newPrice);
20)       System.out.println(prt.price+"...."+newPrice);
21)    }
22) }
```

What is the result?
A) 200.0....100.0
B) 400.0....400.0
C) 400.0....100.0
D) Compilation Fails

Answer: C

Explanation:

In the above example,we are passing Product object reference as argument to updatePrice() method and within the method we are changing the state of object. These changes will be reflected to the caller.

Q18. Consider the code

```
1)  public class Test
2)  {
3)     int x;
4)     int y;
5)     public void doStuff(int x,int y)
6)     {
7)        this.x=x;
8)        y=this.y;
9)     }
10)    public void print()
11)    {
12)       System.out.print(x+":"+y+":");
13)    }
14)    public static void main(String[] args)
15)    {
16)       Test t1=new Test();
17)       t1.x=100;
18)       t1.y=200;
19)
20)       Test t2 = new Test();
21)       t2.doStuff(t1.x,t1.y);
22)       t1.print();
23)       t2.print();
24)    }
25) }
```

What is the result?

A) 100:200:100:0:
B) 100:0:100:0:
C) 100:200:100:200:
D) 100:0:200:0:

Answer: A

Explanation:

If we are passing primitive values to a method as argument and if we are changing any changes to the variable inside method then those changes won't be reflected to the caller because a seperate copy will send to the method.

But if we pass object reference as argument to a method and if we are performing any changes to that object in the method body,those changes will be reflected to the caller also.

Q19. Consider the code

```
1)  public class Test
2)  {
3)     private char ch;
4)     public static void main(String[] args)
5)     {
6)        char ch1='a';
7)        char ch2=ch1;
8)        ch2='e';
9)
10)       Test obj1= new Test();
11)       Test obj2=obj1;
12)       obj1.ch='i';
13)       obj2.ch='o';
14)
15)       System.out.println(ch1+":"+ch2);
16)       System.out.println(obj1.ch+":"+obj2.ch);
17)    }
18) }
```

What is the result?

A)
a:e
o:o

B)
e:e
i:o

C)
a:e
i:o

D)
e:e
o:o

Answer: A

Explanation:

As obj1 and obj2 are pointing to the same object,by using one reference if we perform any changes then those changes will be reflected to other references also.

Topic-5: Main Method and Command Line Arguments

Q1. Which one of the following code examples uses valid java syntax?

A)

```
1)  public class Bunny
2)  {
3)     public static void main(String[] args)
4)     {
5)        System.out.println("Bunny");
6)     }
7)  }
```

B)

```
1)  public class Chinny
2)  {
3)     public static void main(String[])
4)     {
5)        System.out.println("Chinny");
6)     }
7)  }
```

C)

```
1)  public class Sunny
2)  {
3)     public void main(String[] args)
4)     {
5)        System.out.println("Sunny");
6)     }
7)  }
```

D)

```
1)  public class Vinny
2)  {
3)     public static void main(String() args)
4)     {
5)        System.out.println("Vinny");
6)     }
```

```
7)  }
```

Answer: A

Explanation:

The valid way of declaring main method is :
public static void main(String[] args)

Q2. Given the code from the Test.java file:

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        System.out.println("Hello "+args[0]);
6)     }
7)  }
```

Which set of commands prints Hello Durga in the console?
A)
javac Test
java Test Durga

B)
javac Test.java Durga
java Test

C)
javac Test.java
java Test Durga

D)
javac Test.java
java Test.class Durga

Answer: C

Explanation:
To compile a java program we have to use javac command with file name
javac Test.java

We can run with Java command and we have to specify .class file name with out any extension

java Test

if we want to pass any command line arguments at runtime we have to pass like

java Test Durga

Here Durga is the command line argument which can be accessed in the program with args[0]

Q3. Consider the code Test.java:

```
1)  public class Test
2)  {
3)     public static void main(int[] args)
4)     {
5)        System.out.println("int[] main: "+args[0]);
6)     }
7)     public static void main(Object[] args)
8)     {
9)        System.out.println("Object[] main: "+args[0]);
10)    }
11)    public static void main(String[] args)
12)    {
13)       System.out.println("String[] main: "+args[0]);
14)    }
15) }
```

and the commands
javac Test.java
java Test 1 2 3

What is the result?
A) int[] main: 1
B) Object[] main: 1
C) String[] main: 1
D) Compilation Fails
E) An Exception raises at runtime

Answer: C

Explanation: Overloading concept is applicable for main method. But JVM will always call String[] argument main method only.

public static void main(String[] args)

Topic-6: Operators and Assignments

Q1. Consider the following code

```
System.out.println("5 + 2 = "+4+3);
System.out.println("5 + 2 = "+(4+3));
```

What is the result?

A) 5 + 2 = 43
5 + 2 = 43
B) 5 + 2 = 7
5 + 2 = 7
C) 5 + 2 = 7
5 + 2 = 43
D) 5 + 2 = 43
5 + 2 = 7
Answer: D

Explanation:
The only overloaded operator in java is + operator. Sometimes it acts as arithmetic addition operator and sometimes it acts as string concatenation operator.If both arguments are number type then it acts as arithmetic addition operator and if atleast one argument is string type then it acts as concatenation operator. Parenthesis will get highest priority in operator precedence.

Q2. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        System.out.println("Result A:"+ 4+5);
6)        System.out.println("Result B:"+ (4)+(5));
7)     }
8)  }
```

What is the output?

A) Result A:45
Result B:45

B) Result A:45
Result B:9

C) Result A:9
Result B:45

D) Result A:9
Result B:9

Answer: A

Explanation:
The only overloaded operator in java is + operator. Sometimes it acts as arithmetic addition operator and sometimes it acts as string concatenation operator.If both arguments are number type then it acts as arithmetic addition operator and if atleast one argument is string type then it acts as concatenation operator. Parenthesis will get highest priority in operator precedence.
(4) simply treated as 4 only.

Q3. Consider the following code

```
1)  class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int x =100;
6)        int a = x++;
7)        int b = ++x;
8)        int c =x++;
9)        int d=(a<b)?(a<c)?a:(b<c)?b:c;
10)       System.out.println(d);
11)    }
12) }
```

What is the result?
A) 100
B) 101
C) 102
D) 103
E) Compilation fails

Answer : E

Explanation:

int d=(a<b)?(a<c)?a:(b<c)?b:c;
It is invalid syntax

Q4. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int x =1;
6)        int y=0;
7)        if(++x >++y)
8)        {
9)           System.out.print("Hello ");
10)       }
11)       else
12)       {
13)          System.out.print("Hi ");
14)       }
15)       System.out.println("Durga "+x+":"+y);
16)    }
17) }
```

What is the output?
A) Hello Durga 1:0
B) Hello Durga 2:1
C) Hi Durga 1:0
D) Hi Durga 2:1

Answer: B

Q5. Consider the code

```
1)  class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        //Assume x value is 9
6)        if(x++<10)
7)        {
8)           System.out.println(x+" Hello India");
```


```
9)        }
10)       else
11)       {
12)          System.out.println(x+" Hello DURGASOFT");
13)       }
14)    }
15) }
```

Assume if x value is 9 then what is the output?

A) 10 Hello India
B) 10 Hello DURGASOFT
C) 9 Hello India
D) Compilation fails

Answer: A

Q6.Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int i =20;
6)        int j =30;
7)        int k = j += i/5;
8)        System.out.println(i+":"+j+":"+k);
9)     }
10) }
```

What is the output?
A) 20:34:34
B) 4:34:34
C) 20:34:20
D) 34:34:34

Answer: A

Q7. Consider the code

```
1)  public class Test
2)  {
3)     public static final int MIN=1;
4)     public static void main(String[] args)
5)     {
6)        int x = args.length;
7)        if(checkLimit(x))
8)        {
9)           System.out.println("OCJA");
10)       }
11)       else
12)       {
13)          System.out.println("OCJP");
14)       }
15)    }
16)    public static boolean checkLimit(int x)
17)    {
18)       return (x>=MIN) ? true : false;
19)    }
20) }
```

And given the commands as :
javac Test.java
java Test

What is the result ?

A) OCJA
B) OCJP
C) Compilation Fails
D) NullPointerException is thrown at runtime

Answer : B

Q8. Given Student class as

```
1)  public class Student
2)  {
3)     int rollno;
4)     String name;
5)     public Student(int rollno,String name)
```

```
6)     {
7)        this.rollno=rollno;
8)        this.name=name;
9)     }
10) }
```

Consider the code

```
Student s1= new Student(101,"Durga");
Student s2= new Student(101,"Durga");
boolean b1= s1==s2;
boolean b2= s1.name.equals(s2.name);
System.out.println(b1+":"+b2);
```

What is the result?
A) true:true
B) true:false
C) false:true
D) false:false

Answer: C

Explanation:
== Operator is always meant for reference comparison(address comparison).
But for String objects equals() method meant for content comparison.
s1==s2; returns false because both references are not pointing to the same object

s1.name.equals(s2.name) returns true because both names are same

Q9. Given the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String s1= "durga";
6)        String s2= new String("Durga");
7)        //line-1
8)        {
9)           System.out.println("Equal");
10)       }
11)       else
12)       {
```

```
13)          System.out.println("Not Equal");
14)       }
15)    }
16) }
```

Which code to be inserted at line-1 to print Equal

A) String s3=s2;
** if(s1==s3)**
B) if(s1.equalsIgnoreCase(s2))
C) String s3=s2;
** if(s1.equals(s3))**
D) if(s1.toLowerCase() == s2.toLowerCase())

Answer: B

Explanation:
== always meant for reference comparison
equals() for string objects meant for content comparison where case is also will be considered.
equalsIgnoreCase() meant for content comparison where case is not important

Q10. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String s1="Durga";
6)        String[] s2={"D","u","r","g","a"};
7)        String s3="";
8)        for(String s :s2)
9)        {
10)          s3=s3+s;
11)       }
12)       boolean b1= (s1==s3);
13)       boolean b2= (s1.equals(s3));
14)       System.out.println(b1+":"+b2);
15)    }
16) }
```

What is the output?
A)true:true
B)true:false
C)false:true
D)false:false

Answer: C
Explanation:
== Operator is always meant for reference comparison(address comparison).
But for String objects equals() method meant for content comparison.

Q11. Consider the Test class

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        if(args[0].equals("Durga")?false:true)
6)        {
7)           System.out.println("Success");
8)        }
9)        else
10)       {
11)          System.out.println("Failure");
12)       }
13)    }
14) }
```

javac Test.java
java Test Durga

What is the output?

A) Success
B) Failure
C) Compilation fails
D) Runtime Exception

Answer: B

Q12. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String s="OCJA";
6)        String result=null;
7)        if(s.equals("JAVA"))
8)        {
9)           result="First Level";
10)       }
11)       else
12)       {
13)          result="Second Level";
14)       }
15)       System.out.println(result);
16)       result=s.equals("OCJA") ? "First Level" : "Second Level";
17)       System.out.println(result);
18)    }
19) }
```

What is the result?

A)
First Level
Second Level

B)
Second Level
First Level

C)
First Level
First Level

D)
Second Level
Second Level

Answer : B

Q13. Consider the code

```
1)  String s="Color";
2)  String result=null;
3)  if(s.equals("Color"))
4)  {
5)     result="Blue";
6)  }
7)  else if(s.equals("Wall"))
8)  {
9)     result="Regular";
10) }
11) else
12) {
13)    result="No Result";
14) }
```

Which code fragment can replace the if block?

A) s.equals("Color")?result="Blue":s.equals("Wall")?result="Regular" : result="No Result";
B) result = s.equals("Color")?"Blue" else s.equals("Wall")? "Regular" : "No Result";
C) result = s.equals("Color")? s.equals("Wall")? "Blue" : "Regular" : "No Result";
D) result = s.equals("Color")? "Blue" : s.equals("Wall")? "Regular" : "No Result";

Answer: D

Q14. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        double discount=0.0;
6)        int quantity=Integer.parseInt(args[0]);
7)        // Line-1
8)     }
9)  }
```

And the given requirements:

If the value of the quantity variable is greater than or equal to 90,then discount=20

If the value of the quantity variable is between 80 and 90 ,then discount=10

Which two code fragments can be independently placed at Line-1 to meet the requirements ?

A)

```
1)  if (quantity >= 90)
2)  {
3)     discount=20;
4)  }
5)  if (quantity > 80 && quantity < 90)
6)  {
7)     discount=10;
8)  }
```

B)

```
discount=(quantity >= 90 ) ? 20 : 0;
discount=(quantity > 80 ) ? 10 : 0;
```

C)

```
discount = (quantity >= 90 ) ? 20 : (quantity > 80) ? 10 : 0;
```

D)

```
1)  if(quantity >= 80 && quantity <90)
2)  {
3)     discount=10;
4)  }
5)  else
6)  {
7)     discount=0;
8)  }
9)  if (quantity >= 90)
10) {
11)    discount=20;
12) }
13) else
14) {
15)    discount=0;
16) }
```

E) discount= (quantity>80) ? 10 :(quantity >=90)?20:0;

Answer : A and C

Topic-7: Flow-Control

Q1. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        //line-1
6)        switch(x)
7)        {
8)           case 10:
9)              System.out.println("Ten");
10)             break;
11)          case 20:
12)             System.out.println("Twenty");
13)             break;
14)       }
15)    }
16) }
```

Which 3 code fragments can be independently inserted at line-1 to print Ten
A) byte x =10;
B) short x =10;
C) String x ="10";
D) long x =10;
E) double x =10;
F) Integer x = new Integer(10);

Answer: A,B,F

Explanation: The only allowed types for switch argument type are: byte,short,int,char and corresponding wrapper classes,String and enum.
As the case labels are integral numbers,we can take only byte,short,Integer.

Q2. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        boolean b = true;//line-1
6)        switch(b)
7)        {
8)           case true://line-2
9)              System.out.print("True");
10)             break;
11)          default:
12)             System.out.print("default");
13)       }
14)       System.out.println("Done");
15)    }
16) }
```

Which of the following changes are required to print TrueDone?
A) Replace line-1 with String b="true";
Replace line-2 with case "true";

B) Replace line-1 with boolean b=1;
Replace line-2 with case 1:
C) remove break statement
D) remove the default section

Answer: A

Explanation: The only allowed types for switch argument type are: byte,short,int,char and corresponding wrapper classes,String and enum.

Q3. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String color="Green";
6)        switch(color)
7)        {
8)           case "Red":
```

```
9)              System.out.println("Red");
10)         case "Blue":
11)             System.out.println("Blue");
12)                 break;
13)         case "Green":
14)             System.out.println("Green");
15)             break;
16)         default:
17)             System.out.println("Default");
18)       }
19)
20)    }
21)  }
```

What is the output?

A) Red
Blue
B) Green
Default
C) Default
D) Green

Answer: D

Explanation: If we pass String type as argument to switch statement,jvm will identify matched case label with equals() method which is meant for content comparison.

Q4. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int x =10;
6)        int y =20;
7)        switch(x+1)//Line-1
8)        {
9)           case 10:
10)             System.out.println(10);
11)          case y://Line-2
12)             System.out.println(20);
13)             break;
14)          case 11:
```

```
15)             System.out.println(11);
16)         }
17)     }
18) }
```

What is the result?
A) 11
B) Compilation Fails at Line-1
C) Compilation Fails at Line-2
D) Compilation Fails at Line-1 and Line-2

Answer: C

Explanation: For the switch argument we can take any expression but it should returns a single value.
As the case label compulsory we should take constant values.i.e variables are not allowed as case label.

Q5. Which of the following is true about switch statement?

A) It should contain the default section
B) The break statement, at the end of each case block is mandatory
C) Its case lable literals can be changed at runtime
D) Its expression must evaluate a single value
E) boolean type allowed for switch argument type.

Answer : D

Explanation:
default section is optional
break statement is not mandatory for every case block
case labels must be constants and we cannot change at runtime.
Switch statement expression must evaluate a single value

Q6. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int i =5;
6)        while(isAvailable(i))
7)        {
```

```
8)          System.out.print(i);//Line-1
9)          //Line-2
10)       }
11)    }
12)    public static boolean isAvailable(int i)
13)    {
14)       return i-- > 0 ? true : false;// Line-3
15)    }
16)  }
```

Which modification enables the code to print 54321

A) Replace Line-1 with System.out.print(--i);
B) At Line-2, insert i--;
C) Replace Line-1 with --i; and , at Line-2 insert System.out.print(i);
D) Replace Line-3 with return (i> 0) ? true : false;

Answer : B

Q7. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int[] x= {1,2,3,4};
6)        int i =0;
7)        do
8)        {
9)           System.out.print(x[i]+" ");
10)          i++;
11)       }
12)       while (i<x.length -1);
13)    }
14)  }
```

What is the output?
A) 1 2 3 4
B) 1 2 3
C) Compilation Fails
D) IndexOutOfBoundsException

Answer: B

Q8. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int x =5;
6)        do
7)        {
8)           System.out.print(x-- +" ");
9)        }
10)       while (x==0);
11)    }
12) }
```

What is the result?

A) 5 4 3 2 1 0
B) 5 4 3 2 1
C) 4 2 1
D) 5
E) Nothing is printed

Answer: D

Q9. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int[][] x = new int[2][4];
6)        x[0]= new int[]{2,4,6,8};
7)        x[1]= new int[]{2,4};
8)        for(int[] x1: x)
9)        {
10)          for(int x2 :x1)
11)          {
12)             System.out.print(x2+" ");
13)          }
14)          System.out.println();
15)       }
16)    }
```

```
17) }
```

What is the output?

A)
2 4 6 8
2 4
B)
2 4
2 4
C) Compilation Fails
D) ArrayIndexOutOfBoundsException

Answer: A

Q10. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int[]  data={10,20,30,40,50,30};
6)        int k= 30;
7)        int count=0;
8)        for(int x : data)
9)        {
10)          if( x!= k)
11)          {
12)             continue;
13)             count++;
14)          }
15)       }
16)       System.out.println(count);
17)    }
18) }
```

What is the result?

A) 0
B) 1
C) 2
D) Compilation Fails

Answer: D

Explanation:
After break or continue,we cannot take any statement directly. otherwise we will get compiletime error saying unreachable statement.

In the above code if we take count++; outside of if block then output is 2

Q11. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int wd = 0;
6)        String[] days={"sun","mon","wed","sat"};
7)        for(String s : days)
8)        {
9)           switch(s)
10)          {
11)             case "sat":
12)                  case "sun":
13)                wd -= 1;
14)                break;
15)             case "mon":
16)                wd++;
17)             case "wed":
18)                wd += 2;
19)          }
20)       }
21)       System.out.println(wd);
22)   }
23) }
```

What is the output?
A) 3
B) 4
C) -1
D) Compilation Fails

Answer: A

Q12. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int[] x= {1,2,3,4,5};
6)        for(yyy)
7)        {
8)           System.out.print(x[i]);
9)        }
10)    }
11) }
```

Which option can replace yyy to enable the code to print 135?

A) int i=0;i<=4;i++
B) int i=0;i<5;i+=2
C) int i=1;i<=5;i++
D) int i=1;i<5;i+=2

Answer: B

Q13. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String[][] colors=new String[2][2];
6)        colors[0][0]="red";
7)        colors[0][1]="blue";
8)        colors[1][0]="green";
9)        colors[1][1]="yellow";
10)    }
11) }
```

Which code fragment prints red:blue:green:yellow:?
A)

```
1)  for(int i=1;i<2;i++)
2)  {
3)     for(int j =1; j<2;j++)
```

```
4)    {
5)       System.out.print(colors[i][j]+":");
6)    }
7)  }
```

B)

```
1)  for(int i=0;i<2;i++)
2)  {
3)     for(int j =0; j<i;++j)
4)     {
5)        System.out.print(colors[i][j]+":");
6)     }
7)  }
```

C)

```
1)  for(String c: colors)
2)  {
3)     for(String s: c)
4)     {
5)        System.out.print(s+":");
6)     }
7)  }
```

D)

```
1)  for(int i=0;i<2;)
2)  {
3)     for(int j =0; j<2;)
4)     {
5)        System.out.print(colors[i][j]+":");
6)        j++;
7)     }
8)     i++;
9)  }
```

Answer: D

Q14. Consider the following code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int[] s={10,20,30};
6)        int size=3;
7)        int i =0;
8)        /* Line-1 */
9)        System.out.println("The Top Element:"+ s[i]);
10)    }
11) }
```

Which code fragment inserted at line-1 ,prints The Top Element:30 ?

A)

```
1)  do
2)  {
3)     i++;
4)  }
5)  while ( i>= size );
```

B)

```
1)  while(i<size)
2)  {
3)     i++;
4)  }
```

C)

```
1)  do
2)  {
3)     i++;
4)  }
5)  while (i<size-1);
```

D)

```
1)  do
2)  {
3)     i++;
4)  }
5)  while (i<= size);
```

E)

```
1)  while(i<= size-1)
2)  {
3)     i++;
4)  }
```

Answer: C

Q15. Consider the following code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int n[];
6)        n= new int[2];
7)        n[0]=100;
8)        n[1]=200;
9)
10)       n = new int[4];
11)       n[2]=300;
12)       n[3]=400;
13)
14)       for(int x : n)
15)       {
16)          System.out.print(" "+x);
17)       }
18)    }
19) }
```

What is the result?

A) 100 200 300 400
B) 0 0 300 400

C) Compilation Fails
D) An exception thrown at runtime

Answer : B
Explanation: Once we creates an array, every element by default initialized with default values. Based on our requirement we can override default values.

Q16. Consider the following code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int[][] n= {{1,2},{3,4}};
6)        for(int i =n.length-1;i>=0;i--)
7)        {
8)           for(int x:n[i])
9)           {
10)             System.out.print(x);
11)          }
12)       }
13)    }
14) }
```

What is the result?
A) 1234
B) 3412
C) 4321
D) 2143
Answer : B

Q17. Given the code fragment:

int[] x ={10,20,30,40,50};

And the given requirements:

1. Process all the elements of the array in the order of entry.
2. Process all the elements of the array in the reverse order of entry.
3. Process alternating elements of the array in the order of entry.

Which two statements are true?

A) Requirements 1,2 and 3 can be implemented by using the enhanced for loop
B) Requirements 1,2 and 3 can be implemented by using the standard for loop
C) Requirements 2 and 3 CANNOT be implemented by using the standard for loop
D) Requirement 1 can be implemented by using the enhanced for loop
E) Requirement 3 CANNOT be implemented by using either the enhanced for loop or the standard for loop.

Answer: B,D

Q18. Given the following array:

int[] x = {10,20,30,40,50};

Which two code fragments independently print each element of this array?

A)

```
1)  for(int i : x)
2)  {
3)    System.out.print(x[i]+" ");
4)  }
```

B)

```
1)  for(int i : x)
2)  {
3)    System.out.print(i+" ");
4)  }
```

C)

```
1)  for(int i=0 : x)
2)  {
3)    System.out.print(x[i]+" ");
4)    i++;
5)  }
```

D)

```
1)  for(int i=0 ; i < x.length; i++)
2)  {
3)    System.out.print(i+" ");
4)  }
```

E)

```
1)  for(int i=0 ; i < x.length; i++)
2)  {
3)    System.out.print(x[i]+" ");
4)  }
```

F)

```
1)  for(int i=1 ; i < x.length; i++)
2)  {
3)    System.out.print(x[i]+" ");
4)  }
```

Answer : B and E

Explanation:
Standard for loop is index based where as Enhanced for loop is Element based.

Q19. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String[] s={"A","B","C","D"};
6)        for(int i =0; i<s.length;i++)
7)        {
8)           System.out.print(s[i]+" ");
9)           if (s[i].equals("C"))
10)          {
11)             continue;
12)          }
13)          System.out.println("Done");
14)          break;
15)       }
```

```
16)    }
17) }
```

What is the result?

A) A B C D Done
B) A B C Done
C) A Done
D) Compilation fails

Answer : C

Q20. Consider the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String[][] s={{"A","B","C"},{"D","E"}};
6)        for(int i =0; i<s.length;i++)
7)        {
8)           for(int j =0; j<s[i].length;j++)
9)           {
10)             System.out.print(s[i][j]+" ");
11)             if(s[i][j].equals("B"))
12)             {
13)                break;
14)             }
15)          }
16)          continue;
17)       }
18)   }
19) }
```

What is the result?

A) A B C D E
B) A B C
C) A B D E
D) Compilation fails

Answer : C

Q21. Consider the following code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int i = 0;
6)        int j = 7;
7)        for( i = 0; i < j-1 ; i= i+2)
8)        {
9)           System.out.print(i+" ");
10)       }
11)
12)    }
13) }
```

What is the result?

A) 0 2 4 6
B) 0 2 4
C) 2 4 6
D) Compilation fails

Answer : B

Q22. Consider the following code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String[][] s= new String[2][];
6)        s[0] = new String[2];
7)        s[1] = new String[5];
8)        int i =97;
9)
10)       for(int a =0;a < s.length; a++)
11)       {
12)          for(int b =0; b< s.length; b++)
13)          {
14)             s[a][b]=""+i;
15)             i++;
16)          }
```

```
17)       }
18)       for( String[] s1: s)
19)       {
20)          for (String s2 : s1)
21)          {
22)             System.out.print(s2+" ");
23)          }
24)          System.out.println();
25)       }
26)    }
27) }
```

What is the result ?

A)
97 98
99 100 null null null

B)
97 98
99 100 101 102 103

C) Compilation fails

D) NullPointerException is thrown at Runtime

E) ArrayIndexOutOfBoundsException is thrown at Runtime

Answer: A

Q23.
Consider the following code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        int i =1;
6)        int total=0;
7)        while(++i<=5)
8)        {
9)           if(i%2==0)
10)          {
```


```
11)             continue;
12)           }
13)           total=total+i;
14)       }
15)       System.out.println(total);
16)    }
17) }
```

What is the result?

A) 7
B) 8
C) 9
D) 10

Answer : B

Explanation: We can use continue inside loops to skip current iteration and continue for the next iteration.

Topic-8: Declarations and Access Modifiers

Part-1: import statement and packages
Part-2: Access Modifiers(public,private,protected and default)
part-3: abstract modifier and interfaces

Q1. Given the code fragment from 3 files

SalesMan.java:

```
1)  package sales;
2)  public class SalesMan
3)  {
4)  }
```

Product.java:

```
1)  package sales.products;
2)  public class Product
3)  {
4)  }
```

Market.java:

```
1)  package market;
2)  //Line-1
3)  public class Market
4)  {
5)     SalesMan sm;
6)     Product p;
7)  }
```

Which code fragment when inserted at Line-1, enables the code to compile?

A) import sales.*;
B) import java.sales.products.*;
C) import sales;
** import sales.products;**
D) import sales.*;

import products.*;
E) import sales.*;
import sales.products.*;

Answer : E

Explanation:
Whenever we are importing a package, all classes and interfaces present in that package are by default available but not sub package classes. Hence to use sub package class, compulsory we should write import statement until sub package level.

Q2. Given the code fragments:

A.java:

```
1)  package pack1;
2)  public class A
3)  {
4)  }
```

B.java:

```
1)  package pack1.pack2;
2)  //Line-1
3)  public class B
4)  {
5)     public void m1()
6)     {
7)        A a = new A();
8)     }
9)  }
```

C.java:

```
1)  package pack3;
2)  //Line-2
3)  public class C
4)  {
5)     public static void main(String[] args)
6)     {
7)        A a = new A();
8)        B b = new B();
9)     }
```



```
10) }
```

Which modifications enables the code to compile?

A) Replace Line-1 with:
import pack1.A;

Replace Line-2 with:
import pack1.A;
import pack1.pack2.B;

B) Replace Line-1 with:
import pack1;

Replace Line-2 with:
import pack1;
import pack1.pack2;

C) Replace Line-1 with:
import pack1.A;

Replace Line-2 with:
import pack1.*;

D) Replace Line-1 with:
import pack1.*;

Replace Line-2 with:
import pack1.pack2.*;

Answer: A

Explanation:
Whenever we are importing a package, all classes and interfaces present in that package are by default available but not sub package classes. Hence to use sub package class, compulsory we should write import statement until sub package level.

Q3. Which of the following code fragments are valid?

A)

```
1)  public abstract class Test
2)  {
3)     public void m1();
4)     public void m2();
5)  }
```

B)

```
1)  public abstract class Test
2)  {
3)     public abstract void m1();
4)     public void m2();
5)  }
```

C)

```
1)  public abstract class Test
2)  {
3)     public abstract void m1();
4)     public void m2(){}
5)  }
```

D)

```
1)  public abstract class Test
2)  {
3)     public abstract void m1(){}
4)     public abstract void m2(){}
5)  }
```

Answer: C

Explanation:
If we dont want to provide implementation then compulsory we should delcare that method as abstract.

Q4. You are asked to develop a program for a shopping application, and you are given the following information:

The application must contains the classes Book,JavaBook and PythonBook.
The Book class is the super class of other 2 classes.

The int calculatePrice(Book b) method calculates the price of the Book.
The void printBook(Book b) method prints the details of the Book.

Which definition of the Book class adds a valid layer of abstraction to the class hierarchy?

A)

```
1)  public abstract class Book
2)  {
3)     public abstract int calculatePrice(Book b);
4)     public void printBook(Book b){}
5)  }
```

B)

```
1)  public abstract class Book
2)  {
3)     public int calculatePrice(Book b);
4)     public void printBook(Book b);
5)  }
```

C)

```
1)  public abstract class Book
2)  {
3)     public int calculatePrice(Book b);
4)     public final void printBook(Book b){}
5)  }
```

D)

```
1)  public abstract class Book
2)  {
3)     public abstract int calculatePrice(Book b){}
4)     public abstract void printBook(Book b){}
5)  }
```

Answer: A

Explanation:
If we dont want to provide implementation then compulsory we should delcare that method as abstract.

Q5. Consider the code

A.java:

```
1)  package pack1;
2)  public class A
3)  {
4)     int p;
5)     private int q;
6)     protected int r;
7)     public int s;
8)  }
```

Test.java:

```
1)  package pack2;
2)  import pack1.A;
3)  public class Test extends A
4)  {
5)     public static void main(String[] args)
6)     {
7)        A obj= new Test();
8)     }
9)  }
```

Which statement is true?

A) By using obj, we can access p and s
B) By using obj, we can access only s
C) By using obj, we can access r and s
D) By using obj, we can access p,r and s

Answer: B

Explanation:

private members can be accessed within the class only and from outside of the class we cannot access.

default members can be accessed within the package only and from outside of the package we cannot access.

protected members can be accessed within the package anywhere and from outside package only in child classes and we should use child class reference only. To access protected members we cannot use parent class reference from outside of package.

public members can be accessed anywhere and no restrictions.

Q6. Given the content of 3 files

X.java:

```
1)  public class X
2)  {
3)     public void a(){}
4)     int a;
5)  }
```

Y.java:

```
1)  public class Y
2)  {
3)     private int doStuff()
4)     {
5)        private int i =100;
6)        return i++;
7)     }
8)  }
```

Z.java:

```
1)  import java.io.*;
2)  package pack1;
3)  class Z
4)  {
5)     public static void main(String[] args)throws IOException
6)     {
```


```
7)     }
8)  }
```

Which Statement is true?

A) Only X.java file compiles successfully
B) Only Y.java file compiles successfully
C) Only Z.java file compiles successfully
D) Only X.java and Y.java files compile successfully
E) Only Y.java and Z.java files compile successfully
F) Only X.java and Z.java files compile successfully

Answer: A

Explanation:
Local variables cannot be declared as private.
The first non comment statement should be package statement if it is available in any java source file.

Q7. Consider the code

```
1)  interface Interf
2)  {
3)     public void m1();
4)     public void m2();
5)  }
6)  class A implements Interf
7)  {
8)     public void m1(){}
9)  }
```

Which of the following changes individually will compile the code successfully?

A) insert public void m2(){} inside class A
B) declare class A as abstract
C) insert public void m2(); inside class A
D) No Changes are required

Answer: A and B

Explanation:
Whenever we are implementing an interface,compulsory we should provide implementation for every method,otherwise we have to declare class as abstract. Then next level child classes are responsible to provide implementation.

Q8. Consider the code

```
1)  interface Writable
2)  {
3)     public void writeBook();
4)     public void setBookMark();
5)  }
6)  abstract class Book implements Writable //Line-1
7)  {
8)     public void writeBook(){}
9)     //Line-2
10) }
11) class EBook extends Book  //Line-3
12) {
13)    public void writeBook(){}
14)    //Line-4
15) }
```

And given the code Fragment:

```
Book b1= new EBook();
b1.writeBook();
```

Which option enables the code to compile?

A) Replace the code fragment at Line-3 with :
abstract class EBook extends Book

B) Replace the code fragment at Line-1 with :
class Book implements Writable

C) At Line-2 insert
public abstract void setBookMark();

D) At Line-4 insert:
public void setBookMark(){}

Answer : D

Explanation:
Whenever we are implementing an interface,compulsory we should provide implementation for every method,otherwise we have to declare class as abstract. Then next level child classes are responsible to provide implementation.

Topic-9: OOPs

UNIT-5:OOPS Practice Questions

Exam Objectives:

1. Compare and contrast the features and components of Java such as: object orientation, encapsulation, etc.
2. Describe inheritance and its benefits
3. Create methods with arguments and return values; including overloaded methods
4. Apply encapsulation principles to a class
5. Develop code that makes use of polymorphism; develop code that overrides methods; differentiate between the type of a reference and the type of an object
6. Create and overload constructors; differentiate between default and user defined constructors
7. Use super and this to access objects and constructors
8. Determine when casting is necessary

Q1. Given the following classes:

```
1)  public class Employee
2)  {
3)     public int salary;
4)  }
5)  public class Manager extends Employee
6)  {
7)     public int budget;
8)  }
9)  public class Director extends Manager
10) {
11)    public int stockOptions;
12) }
```

And given the following main method:

```
1)  public static void main(String[] args)
2)  {
3)     Employee e = new Employee();
4)     Manager m = new Manager();
```

```
5)     Director d = new Director();
6)     //Line 1
7)  }
```

Which two options fail to compile when placed at Line 1 of the main method?

A. e.salary=50_000;
B. d.salary=80_000;
C. e.budget=2_00_000;
D. m.budget=1_00_000;
E. m.stockOptions=500;
F. d.stockOptions=1_000;

Answer: C and E
Explanation:
By using child class reference we can access both Parent and child class memebers.
But by using Parent reference we can access only Parent class members and we cannot access child specific members.

Q2. Given:

```
1)  public class Test
2)  {
3)     public static void sum(Integer x,Integer y)
4)     {
5)        System.out.println("Integer sum is:"+(x+y));
6)     }
7)     public static void sum(double x,double y)
8)     {
9)        System.out.println("double sum is:"+(x+y));
10)    }
11)    public static void sum(float x,float y)
12)    {
13)       System.out.println("float sum is:"+(x+y));
14)    }
15)    public static void sum(int x,int y)
16)    {
17)       System.out.println("int sum is:"+(x+y));
18)    }
19)    public static void main(String[] args)
20)    {
21)       sum(10,20);
22)       sum(10.0,20.0);
```


```
23)    }
24) }
```

What is the result?

A.
int sum is:30
double sum is:30.0

B.
int sum is:30
float sum is:30.0

C.
Integer sum is:30
double sum is:30.0

D.
Integer sum is:30
float sum is:30.0

Answer: A

Explanation:
In overloading exact argument type will get highest priority.
sum(10,20);===>both arguments are int type hence sum(int x,int y) will be executed.
sum(10.0,20.0);==>Both arguments are double type and hence sum(double x,double y) will be executed.

Q3. Given:

```
1)  class A
2)  {
3)     public void test()
4)     {
5)        System.out.print("A");
6)     }
7)  }
8)  class B extends A
9)  {
10)    public void test()
11)    {
12)       System.out.print("B");
13)    }
14) }
15) public class C extends A
16) {
```

```
17)   public void test()
18)   {
19)      System.out.print("C");
20)   }
21)   public static void main(String[] args)
22)   {
23)      A a1 = new A();
24)      A a2 = new B();
25)      A a3 = new C();
26)      a1.test();
27)      a2.test();
28)      a3.test();
29)   }
30) }
```

What is the output?

A. AAA
B. ABC
C. AAB
D. ABA

Ans: B

Explanation:
In overriding method resolution is always takes care by jvm based on runtime object but not based on reference type.

Q4.Given the code fragment:

```
1)  abstract class Parent
2)  {
3)     protected void resolve()//Line-1
4)     {
5)     }
6)     abstract void rotate();//Line-2
7)  }
8)  class Child extends Parent
9)  {
10)    void resolve()//Line-3
11)    {
12)    }
13)    protected void rotate()//Line-4
```

```
14)   {
15)   }
16) }
```

Which two modifications, made independently, enable the code to compile?

A. Make that method at Line-1 public
B. Make that method at Line-2 public
C. Make that method at Line-3 public
D. Make that method at Line-3 protected
E. Make that method at Line-4 public

Answer: C,D

Explanation: While overriding, we cannot reduce the scope of access modifier, but we can increase the scope. Parent class resolve() method is protected and hence in child class overriding method should be either protected or public.

==========

3 Mantras of Object TypeCasting:

A b = (C) d;

Mantra-1 (At Compile Time):

The type of 'd' and 'C' must have some relationship (either parent to child or child to parent or same type), otherwise we will get compile time error.

Mantra-2 (At Compile Time):

'C' must be either same or derived type of 'A', otherwise we will get compile time error

Mantra-3 (At Runtime Time):

The underlying original object type of 'd' must be either same or derived type of 'C', otherwise we will get runtime exception saying : ClassCastException

==========

Q5. Given:
Base.java:

```
1)  class Base
2)  {
3)     public void test()
4)     {
5)        System.out.println("Base");
6)     }
7)  }
```

DerivedA.java:

```
1)  class DerivedA extends Base
2)  {
3)     public void test()
4)     {
5)        System.out.println("DerivedA");
6)     }
7)  }
```

DerivedB.java

```
1)  class DerivedB extends DerivedA
2)  {
3)     public void test()
4)     {
5)        System.out.println("DerivedB");
6)     }
7)
8)     public static void main(String[] args)
9)     {
10)       Base b1= new DerivedB();
11)       Base b2= new DerivedA();
12)       Base b3= new DerivedB();
13)       b1=(Base)b3;
14)       Base b4=(DerivedA)b3;
15)       b1.test();
16)       b4.test();
17)    }
18)
19) }
```

What is the result?
A.
Base
DerivedA

B.
Base
DerivedB

C.
DerivedB
DerivedB

D.
DerivedB
DerivedA

E. A ClassCastException is thrown at runtime

Answer: C

Explanation:
In the above example test() method is overridden.
In overriding method resolution is always based on runtime object.
b1 and b4 references pointing to the same DerivedB object and hence DerivedB method will be executed.

Q6. Given the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        Short s1=200;
6)        Integer s2=400;
7)        Long  s3=(long)s1+s2;  //Line-1
8)        String s4=(String)(s3*s2);// Line-2
9)        System.out.println(s3);
10)    }
11) }
```

What is the result?

A. 600
B. Compilation Fails at Line-1
C. Compilation Fails at Line-2
D. A ClassCastException is thrown at Line-1
E. A ClassCastException is thrown at Line-2
Answer: C

Explanation:
We cannot convert long type of String. At Line-2 we will get compile time error saying incompatible types: long cannot be converted to String

Q7. Which three statements describe the object oriented features of the java language?

A. Objects cannot be reused
B. A sub class can inherit from a super class
C. Objects can share behaviors with other objects
D. A package must contains more than one class
E. Object is the root class of all other objects
F. A main method must be declared in every class

Answer: B,C,E
Explanation:
Objects can be reused.
A sub class can inherit all variables and methods from a super class.
Objects can share behaviors with other objects
A package can contains any number of classes.
Object is the root class of all other objects.
Every class not required to contain main method.

Q8. What is the name of the Java concept that uses access modifiers to protect variables and hide them within a class?

A. Encapsulation
B. Inheritance
C. Abstraction
D. Instantiation
E. Polymorphism

Ans: A
Explanation: This concept is data hiding, Which is the part of encapsulation.

Q9. Which statement best describes encapsulation?

A. Encapsulation ensures that classes can be designed so that only certain fields and methods of an object are accessible from other objects

B. Encapsulation ensures that classes can be designed so that their methods are inheritable

C. Encapsulation ensures that classes can be designed with some fields and methods declared as abstract.

D. Encapsulation ensures that classes can be designed so that if a method has an argument X,any subclass of X can be passed to that methods.

Answer: A

Explanation:
Encapsulation ensures that classes can be designed so that only certain fields and methods of an object are accessible from other objects, so that we can achieve security.

Q10. Which two are benefits of polymorphism?

A. Faster Code at Runtime
B. More efficient Code at Runtime
C. More Dynamic Code at Runtime
D. More Flexible and Reusable Code at Runtime
E. Code that is protected from extension by other classes

Answer: C,D

Explanation:
The biggest advantage of polymorphism is flexibility. We can use same method name for different arguments(overloaded methods).
We can provide different implementations for the same method in parent and child classes(method overriding).
If the same method available in parent and child classes(overriding) then based on runtime object the corresponding method will be executed (Dynamic Method Dispatch)

Q11. Which three statements are true about the structure of a Java class?

A) public class should compulsory contains main method
B) A class can have only one private constructor
C) A method can have the same name as variable

D) A class can have overloaded static methods
E) The methods are mandatory components of a class
F) The fields need not be initialized before use.

Answer: C, D,F

Explanation:
public class not required to contain main method
A class can have any number of private constructors
A method can have the same name as variable
A class can have overloaded static methods
Methods are not mandatory components of class.
The fields need not be initialized before use

Test.java:

```
1)  class Test
2)  {
3)     static int x =10;
4)     public static void x()
5)     {
6)        System.out.println("Hello");
7)     }
8)     public static void main(String[] args)
9)     {
10)       System.out.println(x);
11)       x();
12)    }
13) }
```

Output:
10
Hello

In the above we are using x as variable and method.

Q12. Given the following code fragment:

```
1)  public class Rectangle
2)  {
3)     private double length;
4)     private double height;
5)     private double area;
```

```
6)    public void setLength(double length)
7)    {
8)       this.length=length;
9)    }
10)   public void setHeight(double height)
11)   {
12)      this.height=height;
13)   }
14)   public void setArea()
15)   {
16)      area=length*height;
17)   }
18) }
```

Which two changes would encapsulation this class and ensure that the area field is always equal to length*height,whenever Rectangle class is used?

A. Change the area field to public
B. Change the setArea() method to private?
C. Call the setArea() method at the beginning of the setLength() method
D. Call the setArea() method at the end of the setLength() method
E. Call the setArea() method at the beginning of the setHeight() method
F. Call the setArea() method at the end of the setHeight() method

Answer: B,F

Q13. Given the following two classes:

```
1)  public class Customer
2)  {
3)     ElectricAccount acct=new ElectricAccount();
4)     public void useElectricity(double kwh)
5)     {
6)        acct.addKwh(kwh);
7)     }
8)  }
9)  public class ElectricAccount
10) {
11)    private double kwh;
12)    public double rate=0.09;
13)    private double bill;
14)    //Line-1
15) }
```

How should you write methods in ElectricAccount class at Line-1 so that the member variable bill is always equal to the value of the member variable kwh multiplied by the member variable rate?

Any amount of electricity used by Customer (represented by an instance of the Customer class) must contribute to the Customer's bill(represented by member variable bill) through the method useElectricity() method. An instance of the customer class should never be able to tamper with or decrease the value of the member variable bill?

A.

```
1)  public void addKwh(double kwh)
2)  {
3)     this.kwh+=kwh;
4)     this.bill=this.kwh*this.rate;
5)  }
```

B.

```
1)  public void addKwh(double kwh)
2)  {
3)     if(kwh>0)
4)     {
5)        this.kwh+=kwh;
6)        this.bill=this.kwh*this.rate;
7)     }
8)  }
```

C.

```
1)  private void addKwh(double kwh)
2)  {
3)     if(kwh>0)
4)     {
5)        this.kwh+=kwh;
6)        this.bill=this.kwh*this.rate;
7)     }
8)  }
```

D.

```
1)  public void addKwh(double kwh)
2)  {
3)     if(kwh>0)
4)     {
5)        this.kwh+=kwh;
6)        setBill(this.kwh)
7)     }
8)  }
9)  public void setBill(double kwh)
10) {
11)    bill=kwh*rate;
12) }
```

Answer: C

Explanation:
Any amount of electricity used by Customer(represented by an instance of the Customer class) must contribute to the Customer's bill(represented by member variable bill) through the method useElectricity() method.

means no one is allowed to call addKwh() method directly, shoulbe be private

An instance of the customer class should never be able to tamper with or decrease the value of the member variable bill

if we pass -ve value for kwh then we can decrease the bill. To prevent this we should check kwh>0

Topic-10: Constructors

Practice Questions for Constructors:

Q1. Given:

```
1)  class Vehicle
2)  {
3)     String type="4w";
4)     int maxSpeed=120;
5)     Vehicle(String type,int maxSpeed)
6)     {
7)        this.type=type;
8)        this.maxSpeed=maxSpeed;
9)     }
10) }
11) class Car extends Vehicle
12) {
13)    String trans;
14)    Car(String trans)
15)    {
16)          //Line-1
17)       this.trans=trans;
18)    }
19)    Car(String type,int maxSpeed,String trans)
20)    {
21)       super(type,maxSpeed);
22)       this(trans);//Line-2
23)    }
24) }
```

And given the code fragment:

```
Car c1= new Car("Auto");
Car c2= new Car("4w",150,"Manual");
System.out.println(c1.type+".."+c1.maxSpeed+".."+c1.trans);
System.out.println(c2.type+".."+c2.maxSpeed+".."+c2.trans);
```

What is the result?
A. 4w 120 Auto
4w 150 Manual

B. null 0 Auto
4w 150 Manual

C. Compilatio Fails only at Line-1
D. Compilatio Fails only at Line-2
E. Compilatio Fails at both Line-1 and Line-2

Answer: E

Explanation: The first line inside any constructor must be either super() or this() and if we are not taking anything then compiler will always place super() then parent class should compulsory contains no-arg constructor otherwise we will get error.

super() and this() should be only in first line of constructor,otherwise we will get error.

Q2. Given:

```
1)  class CD
2)  {
3)     int r;
4)     CD(int r)
5)     {
6)        this.r=r;
7)     }
8)  }
9)  class DVD extends CD
10) {
11)    int c;
12)    DVD(int r, int c)
13)    {
14)       //line-1
15)    }
16) }
```

And Given the code Fragment:

DVD dvd= new DVD(10,20);

Which code fragment should be inserted at Line-1 to instantiate dvd object successfully?

A. super.r=r;
this.c=c;

B. super(r);
this(c);

C. super(r);
this.c=c;

D. this.c=r;
super(c);

Answer: C

Explanation: The first line inside any constructor must be either super() or this() and if we are not taking anything then compiler will always place super() then parent class should compulsory contains no-arg constructor otherwise we will get error.
Hence in the child class constructor we should write explicitly: super(r); and then remaining initialization this.c=c.

The first line inside any constructor must be either super() or this() and we cannot take both simultaneously.

Q3. Given the code Fragment:

```
1)  public class Employee
2)  {
3)     String name;
4)     boolean contract;
5)     double salary;
6)     Employee()
7)     {
8)        //line-1
9)     }
10)    public String toString()
11)    {
12)       return name+":"+contract+":"+salary;
13)    }
14)    public static void main(String[] args)
15)    {
16)       Employee e = new Employee();
```


```
17)       //Line-2
18)       System.out.println(e);
19)    }
20) }
```

Which 2 modifications, when made independently,enable to the code to print Durga:true:100.0

A. Replace line-2 with
```
e.name="Durga";
e.contract=true;
e.salary=100;
```

B. Replace line-2 with
```
this.name="Durga";
this.contract=true;
this.salary=100;
```

C. Replace line-1 with
```
this.name=new String("Durga");
this.contract= new Boolean(true);
this.salary= new Double(100);
```

D. Replace line-1 with
```
name="Durga";
contract=TRUE;
salary=100.0f;
```

E. Replace line-1 with:
```
this("Durga",true,100)
```
Answer: A and C

Explanation:
Inside main method(static method) we cannot use this and super keywords.

We can assign wrapper object to the primitive and compiler will perform required conversions, which is also known as AutoUnboxing.

Q4. Given:

```
1)  class A
2)  {
3)     public A()
4)     {
5)        System.out.println("A");
6)     }
7)  }
8)  class B extends A
9)  {
10)    public B()
11)    {
12)       //line-1
13)       System.out.println("B");
14)    }
15) }
16) class C extends B
17) {
18)    public C()
19)    {
20)       //line-2
21)       System.out.println("C");
22)    }
23)    public static void  main(String[] args)
24)    {
25)       C c = new C();
26)    }
27) }
```

What is the Result?

A. CBA
B. C
C. ABC
D. Compilation Fails at line-1 and line-2

Answer: C

Explanation: Whenever we are creating child class object then both parent class and child class constructors will be executed,but first parent class constructor followed by child class constructor.

Q5. Given

```
1)  class Vehicle
2)  {
3)     int x;
4)     Vehicle()
5)     {
6)        this(10);// line-1
7)     }
8)     Vehicle(int x)
9)     {
10)       this.x=x;
11)    }
12) }
13) class Car extends Vehicle
14) {
15)    int y;
16)    Car()
17)    {
18)       super();
19)       this(20);//line-2
20)    }
21)    Car(int y)
22)    {
23)       this.y= y;
24)    }
25)    public String toString()
26)    {
27)       return super.x+":"+this.y;
28)    }
29) }
```

And given the code fragment:

Vehicle v= new Car();
System.out.println(v);

What is the result?

A. 10:20
B. 0:20
C. Compilation Fails at Line-1
D. Compilation Fails at Line-2

Answer: D

Explanation:
super() and this() should be only in first line of constructor,otherwise we will get error.

Q6. Given the code fragment:

```
1)  public class Person
2)  {
3)     String name;
4)     int age=25;
5)     public Person(String name)
6)     {
7)        this(); //line-1
8)        setName(name);
9)     }
10)    public Person(String name,int age)
11)    {
12)       Person(name);//Line-2
13)       setAge(age);
14)    }
15)    //setter and getter methods go here
16)    public String show()
17)    {
18)       return name+" "+age;
19)    }
20)    public static void main(String[] args)
21)    {
22)       Person p1= new Person("Durga");
23)       Person p2= new Person("Ravi",50);
24)       System.out.println(p1.show());
25)       System.out.println(p2.show());
26)    }
27) }
```

What is the result?
A. Durga 25
Ravi 50

B. Compilation fails at Line-1
C. Compilation fails at Line-2
D. Compilation Fails at both line-1 and line-2

Answer: D

Explanation:
Person class does not contain no-arg constructor.
We cannot call constructor directly by name
Person(name);//Line-2
this(name);//Valid

Q7. Given:

```
1)  class Animal
2)  {
3)     String type="Canine";
4)     int maxSpeed=60;
5)     Animal(){}
6)     Animal(String type,int maxSpeed)
7)     {
8)        this.type=type;
9)        this.maxSpeed=maxSpeed;
10)    }
11) }
12) class WildAnimal extends Animal
13) {
14)    String bounds;
15)    WildAnimal(String bounds)
16)    {
17)       //line-1
18)    }
19)    WildAnimal(String type,int maxSpeed,String bounds)
20)    {
21)       //line-2
22)    }
23) }
```

And the code fragment:

```
WildAnimal wolf= new WildAnimal("Long");
WildAnimal tiger= new WildAnimal("Feline",80,"Short");
System.out.println(wolf.type+" "+wolf.maxSpeed+" "+wolf.bounds);
System.out.println(tiger.type+" "+tiger.maxSpeed+" "+tiger.bounds);
```

Which 2 modifications enable to the code to print the following output?

Canine 60 Long
Feline 80 Short

A. Replace line-1 with
super();
this.bounds=bounds;
B. Repalce line-1 with
this.bounds=bounds;
super();

C. Replace line-2 with
super(type,maxSpeed);
this(bounds);

D. Repalce line-1 with
this("Canine",60);
this.bounds=bounds;

E. Replace line-2 with
super(type,maxSpeed);
this.bounds=bounds;

Answer: A and E

Explanation: The first line inside any constructor must be either super() or this() and we cannot take these in any other line.

The first line inside any constructor must be either super() or this() and we cannot take both simultaneously.

Q8.

```
1)  class Employee
2)  {
3)     private String name;
4)     private int age;
5)     private int salary;
6)
7)     public Employee(String name,int age)
8)     {
9)        setName(name);
```

```
10)     setAge(age);
11)     setSalary(2000);
12)   }
13)   public Employee(String name,int age,int salary)
14)   {
15)     setSalary(salary);
16)     this(name,age);
17)   }
18)   //getter and setter methods goes here
19)   public void printDetails()
20)   {
21)     System.out.println(name+":"+age+":"+salary);
22)   }
23) }
```

Test.java:

```
1)  class Test
2)  {
3)    public static void main(String[] args)
4)    {
5)      Employee e1= new Employee();
6)      Employee e2= new Employee("Durga",50);
7)      Employee e3= new Employee("Ravi",40,5000);
8)      e1.printDetails();
9)      e2.printDetails();
10)     e3.printDetails();
11)   }
12) }
```

What is the result?

A. Compilation fails in the Employee class

B. null:0:0
Durga:50:0
Ravi:40:5000

C. null:0:0
Durga:50:2000
Ravi:40:5000

D. Compilation Fails in the Test class

E. Compilation Fails in both Test and Employee classes

Answer: E

Explanation: Employee class does not contain no-arg constructor.
Hence Test class won't compile because of the line:

Employee e1= new Employee();

super() and this() should be only in first line of constructor,otherwise we will get error.

Q9. Given the following class:

```
1)  public class CheckingAccount
2)  {
3)     public int amount;
4)     //line-1
5)  }
```

And the given the following main method located in another class:

```
1)  public static void main(String[] args)
2)  {
3)     CheckingAccount acct= new CheckingAccount();
4)     //line-2
5)  }
```

Which 3 pieces of code inserted independently,set the value of amount to 100?

A. At line-2 insert:
amount=100;

B. At line-2 insert:
this.amount=100;

C. At line-2 insert:
acct.amount=100;

D. At line-1 insert:

```
1)    public CheckingAccount()
2)    {
3)       amount=100;
```

```
4)    }
```

E. At line-1 insert:

```
1)    public CheckingAccount()
2)    {
3)       this.amount=100;
4)    }
```

F. At line-1 insert:

```
1)    public CheckingAccount()
2)    {
3)       acct.amount=100;
4)    }
```

Answers: C,D and E

Explanation:
From outside of the class,we can access instance variables by using object reference only and we cannot access directly.
From static area we cannot use this and super otherwise we will get compile time error.
From constructor we can access instance variables directly.

Topic-11: Exception Handling

Q1. Which three are advantages of the Java Exception Mechanism?

A. Improves the program structure because the error handing code is separated from the normal program function.

B. Provides a set of standard exceptions that covers all possible errors

C. Improves the program structure beacuase the programmer can choose where to handle exceptions
D. Imporves the program structure because exceptions must be handled in the method in which they occurred.
E. Allows the creation of new exceptions that are tailored to the particular program being created.

Answer:A,C,E

Explanation:
Java Exception Handling provides special keywords(try,catch,finally,throw,throws). It Improves the program structure because the error handing code is separated from the normal program function.

We can handle exception either within the method where it is raised or in caller methods.It Improves the program structure beacuase the programmer can choose where to handle exceptions

Based on our programming requirement, we can define our own customized exceptions also.

Q2. Which 3 statements are true about exception handling?

A. Only unchecked exceptions can be rethrown
B. All Subclasses of the RuntimeException are recoverable
C. The parameter in catch block is of Throwable type
D. All subclasses of RuntimeException must be caught or declared to be thrown
E. All Subclasses of the Exception except RuntimeException class are checked exceptions
F. All subclasses of the Error class are checked exceptions and are recoverable

Answer: B,C,E

Explanation:

Any Exception can be rethrown
All Exceptions are receverable where as all Errors are non-recoverable
We can pass any Throwable type as the argument to catch block
RuntimeException and its child classes,Error and its child classes are unchecked. Except these the remaining are checked.

Q3. Which two statements are true?

A. Error class is unextendable
B. Error class is extendable
C. Error is a RuntimeException
D. Error is an Exception
E. Error is a Throwable

Ans: B,E

Explanation:
We can create a class by extending Error class. The following syntax is valid

```
1)  class Test extends Error
2)  {
3)  }
```

Hence Error is Extendable.
Error is a the child class of Throwable. Hence Error is a Throwable

Q4. Given the code fragment:

```
1)  String[] s= new String[2];
2)  int i=0;
3)  for(String s1: s)
4)  {
5)     s[i].concat("element"+i);
6)     i++;
7)  }
8)  for(i=0; i<s.length;i++)
9)  {
10)    System.out.println(s[i]);
11) }
```

What is the result?

A. element 0
element 1
B. null element 0
null element 1
C. null
null
D. A NullPointerException is thrown at runtime

Answer: D

Once we creates an Array, every element will be initialized with default values. For the String type Array it is null.
On the null reference if we are calling any method then we will get NullPointerException.
s[i].concat("element"+i);==>This line raised NullPointerException as s[i] internally refers null.

Q5. Given the code fragment:

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String[] names={"Thomas","Bunny","Chinny"};
6)        String[] pwds=new String[3];
7)        int i =0;
8)        try
9)        {
10)          for (String n: names)
11)          {
12)             pwds[i]=n.substring(2,6);
13)             i++;
14)          }
15)       }
16)       catch (Exception e)
17)       {
18)          System.out.println("Invalid Name");
19)       }
20)       for(String p: pwds)
21)       {
22)          System.out.println(p);
23)       }
```

```
24)    }
25)  }
```

What is the result?

A.
Invalid Name
omas
null
null

B.
Invalid Name

C.
Invalid Name
omas

D.
Compilation Fails

Answer: A
Explanation:
Once we creates an array, every element will be initialized with default value. For pwds[] the elements will become all null values.
While processing "Bunny" the following line raises StringIndexOutOfBoundsException, because index 5 is not available.
pwds[i]=n.substring(2,6);

Q6. Given the code fragment:

```
1)  import java.util.*;
2)  public class Test
3)  {
4)     public static void main(String[] args)
5)     {
6)        ArrayList l = new ArrayList();
7)        try
8)        {
9)           while(true)
10)          {
11)             l.add("MyString");
12)          }
```


```
13)      }
14)      catch (RuntimeException e)
15)      {
16)         System.out.println("Caught a RuntimeException");
17)      }
18)      catch (Exception e)
19)      {
20)         System.out.println("Caught an Exception");
21)      }
22)      System.out.println("Ready to use");
23)   }
24)}
```

What is the result?

A. Caught a RuntimeException printed to the console
B. Caught an Exception printed to the console
C. A runtime error is thrown at runtime
D. Ready to use printed to the console
E. The code fails to compile because a throws keyword required

Answer: C

Explanation: If we are keep on adding String object to the ArrayList, at certain point we will get OutOfMemoryError, which is non recoverable.

Q7. Given the code fragment:

```
1)  import java.io.*;
2)  class X
3)  {
4)     public void printFileContent() throws IOException
5)     {
6)        throw new IOException();//Line-1
7)     }
8)  }
9)  public class Test
10){
11)   public static void main(String[] args)//Line-2
12)   {
13)      X x= new X();
14)      x.printFileContent();//Line-3
15)      //Line-4
```

```
16)
17)    }
18) }
```

Which two modifications required to compile code successfully?

A. Replace Line-2 with
public static void main(String[] args) throws Exception

B. Replace Line-3 with:

```
1)    try
2)    {
3)     x.printFileContent();
4)    }
5)    catch (Exception e){}
6)    catch (IOException e){}
```

C. Replace Line-2 with
public static void main(String[] args) throws IOException

D. Replace Line-1 with
throw IOException("Exception Raised");

E. At Line-4 insert throw new IOException();

Answer: A, C

Explanation:
If we are calling a method and that method throws some checked exception, then compulsory caller should handle that checked exception either by try-catch or throws keyword.

In the above example, main method is responsible to handle IOException, as it is calling printFileContent() method. Hence the following are valid modifications to compile the code successfully.

Replace Line-2 with
public static void main(String[] args) throws Exception

Replace Line-2 with
public static void main(String[] args) throws IOException

Q8. Given the code Fragment:

```
1)  public class Test
2)  {
3)     void readCard(int cno) throws Exception
4)     {
5)        System.out.println("Rearding Card");
6)     }
7)     void checkCard(int cno) throws RuntimeException//Line-1
8)     {
9)        System.out.println("Checking Card");
10)    }
11)    public static void main(String[] args)
12)    {
13)       Test t = new Test();
14)       int cardNo=1234;
15)       t.checkCard(cardNo);//Line-2
16)       t.readCard(cardNo);//Line-3
17)    }
18) }
```

What is the result?

A. Checking Card
Reading Card

B. Compilation Fails at Line-1
C. Compilation Fails at Line-2
D. Compilation Fails at Line-3
E. Compilation Fails at Line-2 and Line-3

Answer: D

Explanation: In our code if we are calling a method which throws checked exception compulsory we should handle either by try-catch or by throws keyword,either the same exception or its parent

Q9. Given the following code for the classes MyException and Test:

```
1)  public class MyException extends RuntimeException
2)  {
3)  }
4)
5)  public class Test
6)  {
7)     public static void main(String[] args)
8)     {
9)        try
10)       {
11)          m1();
12)       }
13)       catch (MyException e)
14)       {
15)          System.out.print("A");
16)       }
17)    }
18)    public static void m1()
19)    {
20)       try
21)       {
22)          throw Math.random() > 0.5 ? new Exception():new MyException();
23)       }
24)       catch (RuntimeException e)
25)       {
26)          System.out.println("B");
27)       }
28)    }
29) }
```

What is the result?
A. A
B. B
C. Either A or B
D. AB
E. Compilation Fails

Answer: E

Explanation:

In our code, if there is a chance of raising checked exception ,compulsory we should handle either by try-catch or by throws statement. In the above code, inside m1() there may be a chance of raising Exception, which is checked, but we didnot handled. Hence we will get compile time error.

Topic-12: String and StringBuilder

Practice Questions for String and StringBuilder:

Q1. Given:

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String ta="A";
6)        ta=ta.concat("B");
7)        String tb="C";
8)        ta=ta.concat(tb);
9)        ta.replace('C','D');
10)       ta=ta.concat(tb);
11)       System.out.println(ta);
12)    }
13) }
```

What is the result?

A. ABCD
B. ACD
C. ABCC
D. ABD
E. ABDC

Answer: C

Explanation:
String objects are immutable. Hence, once we creates a string object,we cannot perform any changes in that object. If we are trying to perform any changes with those changes a new object will be created.

Q2. Consider the following code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String str=" ";
```

```
6)        str.trim();
7)        System.out.println(str.equals("")+" "+str.isEmpty());
8)     }
9)  }
```

What is the result?

A. true false
B. true true
C. false true
D. false false

Answer: D

Explanation:
String objects are immutable. Once we creates string object we cannot perform any changes in the existing object. If we are trying to perform any changes with those changes a new object will be created.

str.trim(); A new object will be created and existing object won't be changed.

Q3. Given:

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String s="DURGA SOFT";
6)        int len=s.trim().length();
7)        System.out.println(len);
8)     }
9)  }
```

What is the result?

A. 10
B. 9
C. 8
D. Compilation Fails

Answer: A

Explanation: trim() method will remove spaces present at beginning and end of string but not middle blank spaces.

Q4. Given

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String s="Hello World";
6)        s.trim();
7)        int i1=s.indexOf(" ");
8)        System.out.println(i1);
9)     }
10) }
```

What is the result?

A. An exception is thrown at runtime
B. -1
C. 5
D. 0

Answer: C

Explanation:
String objects are immutable. Once we creates string object we cannot perform any changes in the existing object. If we are trying to perform any changes with those changes a new object will be created.

trim() method will remove spaces present at beginning and end of string but not middle blank spaces.

In the above example, trim() method won't remove the blank space present at 5th index, because it is present at middle of the string.

Q5. Consider the following code:

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String s1="Java";
```


```
6)      String s2= new String("java");
7)      //Line-1
8)      {
9)         System.out.println("Equal");
10)     }
11)     else
12)     {
13)        System.out.println("Not Equal");
14)     }
15)  }
16) }
```

To print "Equal" which code fragment should be inserted at Line-1

A. String s3=s2;
if(s1==s3)

B. if(s1.equalsIgnoreCase(s2))

C. String s3=s2;
if(s1.equals(s3))

D. if(s1.toLowerCase()==s2.toLowerCase())

Answer: B

Explanation:
In String class, equals() method meant for content comparison where case is important. If we want to ignore case then we should go for equalsIgnoreCase() method. == operator always meant for content comparison.

Q6. Given the code fragment:

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        StringBuilder sb=new StringBuilder(5);
6)        String s="";
7)        if(sb.equals(s))
8)        {
9)           System.out.println("Match 1");
10)       }
```

```
11)      else if(sb.toString().equals(s.toString()))
12)      {
13)         System.out.println("Match 2");
14)      }
15)      else
16)      {
17)         System.out.println("No Match");
18)      }
19)   }
20) }
```

What is the result?

A. Match 1
B. Match 2
C. No Match
D. NullPointerException is thrown at runtime

Answer: B

Explanation:
In StringBuilder equals() method is not overridden and hence object class equals() method will be executed. If arguments are different type then equals() method return false. Hence sb.equals(s) returns false.

But String class equals() method meant for content comparison. Hence sb.toString().equals(s.toString()) returns true.

Q7. Given the code fragment:

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        StringBuilder sb1= new StringBuilder("Durga");
6)        String str1=sb1.toString();
7)        //insert code here==>Line-1
8)        System.out.println(str1==str2);
9)     }
10) }
```

Which code fragment, when inserted at Line-1,enables the code to print true?

A. String str2=str1;
B. String str2=new String(str1);
C. String str2=sb1.toString();
D. String str2="Durga";

Answer: A

Explanation: str1==str2 returns true iff both references pointing to the same object

Q8. Which statement will empty the contents of a StringBuilder variable named sb?

A. sb.deleteAll();
B. sb.delete(0,sb.size());
C. sb.delete(0,sb.length());
D. sb.removeAll();

Answer: C

Explanation:
On the StringBuilder object we cannot apply deleteAll(),size() and removeAll() methods.
But we can apply delete() and length() methods.
sb.delete(begin,end) removes all characters from begin index to end-1 index.
sb.delete(0,sb.length()) removes all characters from the StringBuilder.

Q9. Given the following code:

```
1)  class MyString
2)  {
3)     String msg;
4)     MyString(String msg)
5)     {
6)        this.msg=msg;
7)     }
8)  }
9)  public class Test
10) {
11)    public static void main(String[] args)
12)    {
13)       System.out.println("Hello "+ new StringBuilder("Java SE 8"));
14)       System.out.println("Hello "+ new MyString("Java SE 8"));
15)    }
```

```
16) }
```

What is the result?

A.
Hello Java SE 8
Hello MyString@<hashcode>

B.
Hello Java SE 8
Hello Java SE 8

C.
Hello java.lang.StringBuilder@<hashcode>
Hello MyString@<hashcode>

D. Compilation Fails

Answer: A

Explanation:
In StringBuilder class toString() method is overridden for meaningful String representation. But in MyString class toString() method is not overridden and hence object class toString() method will be called which returns the String in the following format: classname@<hashcode_in_hexadecimal_string_form>

Q10. You are developing a banking module. You have developed a class named MaskTest that has a mask method.
Given the code fragment:

```
1)  class MaskTest
2)  {
3)     public static String mask(String creditCard)
4)     {
5)        String x="XXXX-XXXX-XXXX-";
6)        //Line-1
7)     }
8)     public static void main(String[] args)
9)     {
10)       System.out.println(mask("1234-5678-9101-5979"));
11)    }
12) }
```

You must ensure that mask method returns a String that hides all digits of the credit card number except last four digits(and the hyphens that seperate each group of 4 digits)

Which two code fragments should you use at line 1, independently to achieve the requirement?

A.
```
StringBuilder sb=new StringBuilder(creditCard);
sb.substring(15,19);
return x+sb;
```

B.
```
return x+creditCard.substring(15,19);
```

C.
```
StringBuilder sb=new StringBuilder(x);
sb.append(creditCard,15,19);
return sb.toString();
```

D.
```
StringBuilder sb=new StringBuilder(creditCard);
StringBuilder s=sb.insert(0,x);
return s.toString();
```

Answer: B,C

Explanation:
Only in the following 2 cases the digits will be hidden and in remaining cases all digits will be displayed to the end user.

1. return x+creditCard.substring(15,19);
2.
```
StringBuilder sb=new StringBuilder(x);
sb.append(creditCard,15,19);
return sb.toString();
```

Topic-13: Wrapper Classes

Practice Questions for Wrapper Classes:

Q1. Consider the code:

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        Boolean[] b = new Boolean[2];
6)        b[0]=new Boolean(Boolean.parseBoolean("true"));
7)        b[1]= new Boolean(null);
8)        System.out.println(b[0]+".."+b[1]);
9)     }
10) }
```

What is the result?

A. true..false
B. true..null
C. Compilation Fails
D. NullPointerException is thrown at runtime

Answer: A

Explanation:
new Boolean(Boolean.parseBoolean("true")); represents true
new Boolean(null); represents false

Q2. Given:

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        boolean a = new Boolean(Boolean.valueOf(args[0]));
6)        boolean b = new Boolean(args[1]);
7)        System.out.println(a+".."+b);
8)     }
9)  }
```

And given the commands:
javac Test.java
java Test TRUE null

What is the result?

A. true..null
B. true..false
C. false..false
D. true..true

Answer: B

Explanation:
Whenever we are converting String type to boolean type then both case and content are not important. If the argument is case insensitive string of "true" then it is treated as true otherwise it is treated as false.

boolean b = new Boolean("TRUE");==>true
boolean b = new Boolean("null");==>false

Q3. Given the code

```
1)  public class Test
2)  {
3)     public static void main(String[] args)
4)     {
5)        String s1="123";
6)        String s2="TRUE";
7)        Integer i1=Integer.parseInt(s1);
8)        Boolean b1= Boolean.parseBoolean(s2);
9)        System.out.println(i1+".."+b1);
10)
11)       int i2= Integer.valueOf(s1);
12)       boolean b2=Boolean.valueOf(s2);
13)       System.out.println(i2+".."+b2);
14)    }
15) }
```

What is the result?
A.
123..true
123..true

B.
123..true
123..false
C.
123..false
123..true
D. Compilation Fails

Answer: A

Explanation:
Whenever we are converting String type to boolean type then both case and content are not important.
If the argument is case insensitive string of "true" then it is treated as true otherwise it is treated as false.

In the place of primitive we can provide wrapper object and in the place of wrapper object we can provide primitive. All required conversions will be performed automatically by compiler and these automatic conversions are called autoboxing and autounboxing.

```
Integer i1=Integer.parseInt("123"); // 123
Boolean b1= Boolean.parseBoolean("TRUE");//true

int i2= Integer.valueOf("123");//123
boolean b2=Boolean.valueOf("TRUE");//true
```

Topic-14: ArrayList

Practice Questions for ArrayList :

Q1. Given the code fragment:

```
1)  import java.util.*;
2)  class Test
3)  {
4)     public static void main(String[] args)
5)     {
6)        ArrayList<Integer> l = new ArrayList<>();
7)        l.add(1);
8)        l.add(2);
9)        l.add(3);
10)       l.add(4);
11)       l.add(null);
12)       l.remove(2);
13)       l.remove(null);
14)       System.out.println(l);
15)    }
16) }
```

What is the result?

A. [1, 2, 4]
B. NullPointerException is thrown at runtime
C. [1, 2, 4,null]
D. [1, 3, 4,null]
E. [1, 3, 4]
F. Compilation Fails

Answer: A
Explanation:

null insertion is possible for ArrayList.
l.remove(2) will remove the element locating at index:2, which is nothing but 3. Hence the output is :[1,2,4]

Q2. Given the code fragment:

```
1)  import java.util.*;
2)  class Test
3)  {
4)     public static void main(String[] args)
5)     {
6)        List<String> l = new ArrayList<>();
7)        l.add("Sunny");
8)        l.add("Bunny");
9)        l.add("Chinny");
10)       l.add("Bunny");
11)       if(l.remove("Bunny"))
12)       {
13)          l.remove("Vinny");
14)       }
15)       System.out.println(l);
16)    }
17) }
```

What is the result?

A. [Sunny, Chinny, Bunny]
B. [Sunny, Chinny]
C. [Sunny, Bunny, Chinny, Bunny]
D. An exception is thrown at runtime

Answer: A
Explanation:
l.remove("Bunny") removes the first occurrence of "Bunny" and returns true.
l.remove("Vinny") just returns false,because "Vinny" not present in List.
Hence the output is : [Sunny, Chinny, Bunny]

Q3.
Given:

```
1)  import java.util.*;
2)  class Patient
3)  {
4)     String name;
5)     public Patient(String name)
6)     {
7)        this.name=name;
```


```
8)      }
9)   }
10) class Test
11) {
12)    public static void main(String[] args)
13)    {
14)       List l = new ArrayList();
15)       Patient p = new Patient("Ravi");
16)       l.add(p);
17)       //insert code here==>Line-1
18)       if(f>=0)
19)       {
20)          System.out.println("Ravi Found");
21)       }
22)    }
23) }
```

Which code inserted at Line-1 enable the code to print Ravi Found.

A.
int f=l.indexOf(p);
B.
int f=l.indexOf(Patient("Ravi"));
C.
int f=l.indexOf(new Patient("Ravi"));
D.
Patient p1 = new Patient("Ravi");
int f=l.indexOf(p1);

Answer: A

Explanation: Inside indexOf() method, equals() method will be called to check whether the given object is available or not. If the specified object is available then it returns index of that element. If it is not available then it returns -1. In the Patient class, equals() method is not overridden and hence object class equals() method will be called which is always meant for reference comparison.

Q4. Given the code fragment

```
1)  import java.util.*;
2)  class Test
3)  {
4)     public static void main(String[] args)
```



```
5)    {
6)       ArrayList l = new ArrayList();
7)       try
8)       {
9)          while(true)
10)         {
11)            l.add("MyString");
12)         }
13)      }
14)      catch (RuntimeException e)
15)      {
16)         System.out.println("RuntimeException caught");
17)      }
18)      catch (Exception e)
19)      {
20)         System.out.println("Exception caught");
21)      }
22)      System.out.println("Ready to use");
23)   }
24) }
```

What is the result?

A.
RuntimeException caught
Ready to use
B.
Exception caught
Ready to use
C. Compilation Fails
D. A runtime error thrown in the thread main

Answer: D

Explanation: Here we are keep on adding "MyString" object to List and at certain point sufficient memory may not be available. Then we will get OutOfMemoryError which can not be handled by either of the try block. Hence the program will be terminated abnormally by raising runtime error in thread main. 'Exception in thread "main" java.lang.OutOfMemoryError: Java heap space'

Q5. Given the following class declarations

```
public abstract class Animal{}
public interface Hunter{}
public class Cat extends Animal implements Hunter{}
public class Tiger extends Cat{}
```

Which one fails to compile?

A.

```
ArrayList<Animal> l = new ArrayList<>();
l.add(new Tiger());
```

B.

```
ArrayList<Hunter> l = new ArrayList<>();
l.add(new Cat());
```

C.

```
ArrayList<Hunter> l = new ArrayList<>();
l.add(new Tiger());
```

D.

```
ArrayList<Tiger> l = new ArrayList<>();
l.add(new Cat());
```

E.

```
ArrayList<Animal> l = new ArrayList<>();
l.add(new Cat());
```

Answer: D
Explanation:

```
ArrayList<Tiger> l = new ArrayList<>();
```

For this ArrayList we can add either Tiger object or its child class objects.
We cannot add Cat object because it is Parent of Tiger class. Hence we will get compile time error.

Topic-15: Lambda Expressions

Q1. Given the code fragment:
Person.java:

```
1)  public class Person
2)  {
3)     String name;
4)     int age;
5)     public Person(String n,int a)
6)     {
7)        name=n;
8)        age=a;
9)     }
10)    public String getName()
11)    {
12)       return name;
13)    }
14)    public int getAge()
15)    {
16)       return age;
17)    }
18) }
```

Test.java:

```
1)  class Test
2)  {
3)     public static void checkAge(List<Person> list,Predicate<Person> predicate)
4)     {
5)        for (Person p: list)
6)        {
7)           if(predicate.test(p))
8)           {
9)              System.out.print(p.name+" ");
10)          }
11)       }
12)    }
13)    public static void main(String[] args)
```

```
14)  {
15)     List<Person> iList=Arrays.asList(new Person("Durga",45),
16)                                 new Person("Ravi",40),
17)                                 new Person("Shiva",38));
18)     //line-1
19)
20)  }
21) }
```

Which code fragment,when inserted at line-1 enables the code to print Durga?

A. checkAge(iList,()->p.getAge()>40);
B. checkAge(iList,Person p->p.getAge()>40);
C. checkAge(iList,p->p.getAge()>40);
D. checkAge(iList,(Person p)-> {p.getAge()>40;});

Answer: C

Explanation:
For the Predicate,compulsory we should pass some input argument.
Whenever we are specifying the type, we should use Parenthesis.
If we are enclosing body within curly braces, then compulsory we should use return keyword to return value.

Topic-16: Date and Time API

Practice Questions for Date and Time API:

Case-1:

LocalDate date=LocalDate.of(yyyy,mm,dd);
only valid values we have to take for month,year and day

LocalDate dt=LocalDate.of(2012,01,32);==>invalid
LocalDate dt=LocalDate.of(2012,15,28);===>invalid
LocalDate dt=LocalDate.of(2012,7,28);===>valid

Q1. Given the code fragment:

```
1)  import java.time.*;
2)  public class Test
3)  {
4)     public static void main(String[] args)
5)     {
6)        LocalDate dt=LocalDate.of(2012,01,32);
7)        dt.plusDays(10);
8)        System.out.println(dt);
9)     }
10) }
```

What is the result?
A. 2012-02-10
B. 2012-02-11
C. Compilation Fails
D. DateTimeException thrown at runtime

Answer: D

RE:
Exception in thread "main" java.time.DateTimeException: Invalid value for DayOfMonth (valid values 1 - 28/31):
32

LocalDate class parse methods:

LocalDate class contains the following 2 parse methods.

1. public static LocalDate parse(CharSequence text)

Obtains an instance of LocalDate from a text string such as 2007-12-03.
The string must represent a valid date and is parsed using DateTimeFormatter.ISO_LOCAL_DATE.

The methods throws DateTimeParseException - if the text cannot be parsed

2. public static LocalDate parse(CharSequence text,
DateTimeFormatter formatter)
Obtains an instance of LocalDate from a text string using a specific formatter.
The text is parsed using the formatter, returning a date.

The methods throws DateTimeParseException - if the text cannot be parsed

Note: CharSequence is an interface and its implemented classes are String,StringBuffer,StringBuilder etc

LocalDate class format() method:

---

public String format(DateTimeFormatter formatter)
Formats this date using the specified formatter.
This date will be passed to the formatter to produce a string.

Q2. Given the code Fragment:

```
1)  import java.time.*;
2)  import java.time.format.*;
3)  public class Test
4)  {
5)     public static void main(String[] args)
6)     {
7)        String date=LocalDate.parse("2014-05-
    04").format(DateTimeFormatter.ISO_DATE_TIME);
8)        System.out.println(date);
9)     }
10) }
```

What is the result?

A) May 04,2014T00:00:00.000
B) 2014-05-04T00:00:00.000
C) 5/4/14T00:00:00.000
D) An exception is thrown at runtime

Answer: D

Explanation: Here we have only Date value, but we are passing DateTimeFormatter.ISO_DATE_TIME.

RE:
Exception in thread "main" java.time.temporal.UnsupportedTemporalTypeException: Unsupported field: HourOfDay
at java.time.LocalDate.get0(Unknown Source)

Eg:
LocalDateTime dt=LocalDateTime.parse("2014-05-04T13:45:45.000");
String s=dt.format(DateTimeFormatter.ISO_DATE_TIME);
System.out.println(s);

output: 2014-05-04T13:45:45

Q3. Given the code fragment:

```
1)  import java.time.*;
2)  import java.time.format.*;
3)  public class Test
4)  {
5)     public static void main(String[] args)
6)     {
7)        LocalDate date1=LocalDate.now();
8)        LocalDate date2=LocalDate.of(2018,4,15);
9)        LocalDate date3=LocalDate.parse("2018-04-15",
    DateTimeFormatter.ISO_DATE);
10)       System.out.println("date-1:"+date1);
11)       System.out.println("date-2:"+date2);
12)       System.out.println("date-3:"+date3);
13)    }
14) }
```

What is the result?
A.
date-1:2018-04-15
date-2:2018-04-15
date-3:2018-04-15

B.
date-1:04/15/2018
date-2:2018-04-15
date-3:Apr 15,2018

C. Compilation Fails

D. A DateParseException is thrown at runtime

Answer: A

Explanation: Bydefault LocalDate.now(),LocalDate.of() follows the format: yyyy-mm-dd.

Q4. Given the code fragment:

```
1)  import java.time.*;
2)  import java.time.format.*;
3)  public class Test
4)  {
5)     public static void main(String[] args)
6)     {
7)        LocalDateTime dt=LocalDateTime.of(2014,7,31,1,1);
8)        dt.plusDays(30);
9)        dt.plusMonths(1);
10)       System.out.println(dt.format(DateTimeFormatter.ISO_DATE));
11)    }
12) }
```

What is the result?

A. 2014-07-31
B. 07-31-2014
C. 2014-09-30
D. An Exception is thrown at runtime

Answer: A

Explanation: LocalDateTime is an immutable date-time object that represents a date-time.

dt.plusDays(30);
dt.plusMonths(1);

With these new objects will be created and dt is always point to specified date only(2014,7,31,1,1)

Note: LocalDate,LocalTime and LocalDateTime are immutable ojects.

Topic-17: Garbage Collection

Q1. Given:

```
1)  public class MarkList
2)  {
3)     int num;
4)     public static void graceMarks(MarkList obj4)
5)     {
6)        obj4.num+=10;
7)     }
8)     public static void main(String[] args)
9)     {
10)       MarkList obj1= new MarkList();
11)       MarkList obj2=obj1;
12)       MarkList obj3=null;
13)       obj2.num=60;
14)       graceMarks(obj2);
15)    }
16) }
```

How many MarkList instances are created in memory at runtime?

A. 1
B. 2
C. 3
D. 4

Answer: A

Explanation:
In the above program only one object got created, which is pointed by obj1,obj2 and obj4.
obj3 is not pointing to any object.
Hence the number of objects created is only 1.

Q2. Given the code fragment:

```
1)  class Student
2)  {
3)     String name;
4)     int age;
```


```
5)  }
6)  And,
7)  public class Test
8)  {
9)     public static void main(String[] args)
10)    {
11)       Student s1= new Student();
12)       Student s2= new Student();
13)       Student s3= new Student();
14)       s1=s3;
15)       s3=s2;
16)       s2=null;-->line-1
17)    }
18) }
```

Which statement is true?

A. After line-1, three objects eligible for garbage collection
B. After line-1, two objects eligible for garbage collection
C. After line-1, one object eligible for garbage collection
D. After line-1, no object eligible for garbage collection

Answer: C

Explanation:
If an object does not contain any reference variable then it is said to be eligible for Garbage Collection. After Line-1 only one object eligible for GC which was pointed by s1 at the beginning.