Grade - 2 Vaibhav Ramkisan Chopade. J.30

PRACTICE PAPER

शालेय शिक्षा

$\frac{10 + 40 + 12}{20 + 50 + 30} = \frac{62}{100}$

Hollow craft sticker

Centre No. केंद्र क्रमांक

Seat No. बैठक क्रमांक

बारकोड ह्या चौकटीत चिकटवा.

Paste the barcode sticker here.

In Figure अंकात

In Words अक्षरात

Candidate's Signature परीक्षार्थीची सही

Supervisor's Signature पर्यवेक्षकाची सही

Hollow craft sticker

Hollow craft sticker

Hollow craft sticker

बारकोड संबंधी सुचना

१) विद्यार्थ्यांनी प्रथम पर्यवेक्षकांकडुन बारकोड स्टिकर घ्यावे.

२) बारकोड वरील बैठक क्रमांक व विषय बरोबर असल्याची खात्री करुन घ्यावी.

३) बारकोड स्टिकर त्यासाठी दिलेल्या जागेमध्ये घडी पडू न देता चिकटवावा.

४) बारकोडवर अन्य कोणतेही लिखाण करु नये. केल्यास परीक्षेतील गैरमार्गाचा प्रकार मानण्यात येईल.

५) उत्तरपत्रिकेच्या मुखपृष्ठावरील अन्य माहीती, विहीत जागेत लिहावी.

SSC / HSC

फक्त UID कोडसाठी जागा
(पर्यवेक्षक वापरासाठी)

SSC / HSC

Subject/ विषय	
Paper/ पेपर	
Date/ दिनांक	
Language of Answer उत्तर लेखनाची भाषा	

Suppliments Attached		
Main Answer Book	No. of Supplements	Total in Figure
1	+	=

Specific remarks of centre conductor regarding malpractices (In Red Ink)
उमेदवारांकडून घडलेल्या गैरमार्गासंबंधी केंद्र संचालकाचे अभिप्राय (लाल शाईमध्ये)

	Marks in Figure	Marks in Word	Signature	Appt. No.
Examiner				
Moderator				
Chief Moderator				

Q. No.	Examiner	Moderator	Chief Moderator	
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
Total In Figure				
Total Words				

Signature				
Appt. No.				

Q.No.						Q.No.					

प्र. क्र.
Q. No.

Section - A

not given explanation

1) ⇒ d) 25

2) ⇒ d) It has no class name

3) ⇒ c) Compile error

4) ⇒

5) ⇒ d) fixe set of coding rules.

(10)

5) ⇒ Design pattern is ~~software~~ is a general, reusable solutions to a commonly occuring problem within a given content in sofware design.

section B ⇒ (1) singelton design pattern ?

⇒ - singleton is a part of "Gang of four" design pattern and it categorized under creational.

- singleton pattern is a design pattern which restricts a class to instantiate its multiple objects.

- It is nothing but a way of define class.

- It is used where only a single instance of a class is required to control the action throught the execution.

(10)

Q.No. | Q.No.

प्र. क्र.
Q. No.

- A singleton shouldn't have multiple instance in any case and at any cost.
- singleton classes are used for logging, driver objects, caching and thread pool, databases, connections.

Ex

```
class Singleton {
    private static Singleton obj;
    private Singleton () { }
    public static Singleton getInstance()
    {
        if (obj == null)
            obj = new Singleton();
        return obj;
    }
}
```

Q 2) Constructor Chaining

=> constructor chaining is a feature in java that allows to call one constructor from another constructor in the same class. This can be useful for reusing code and making your constructor more readable.

- To chain the constructor we use this() keyword.

Q.No. | Q.No.

ex

```
class chain {
    chain ()
    {
        this (5);
        System.out.println ("This Default");
    }
    chain (int x)
    {
        this (5, 15);
        System.out.println (x);
    }
    chain (int x, int y)
    {
        System.out.println (x*y);
    }
    public static void main (String [] args)
    {
        new Chain ();
    }
}
```

(10)

9) creating JPA entity

→ ① select the JPA project in the project field.

② select the location of the JPA project, src folder in the source folder field

Q.No. | Q.No.

प्र. क्र.
Q. No. 5

③ Enter the name of the package for the entity in the Java package ~~Afield~~ field.

④ Enter the name of the Java class in the class name field.

Q. 8)

ArrayList	LinkedList
1) ArrayList internally uses a dynamic array to store the elements	1) LinkedList internally uses a doubly linked list to store the elements.
2) manipulation is slow	2) manipulation is faster than ArrayList.
3) An ArrayList act class can act as a list only because it implement List only.	3) LinkedList class can act as a list and queue both because it implements List & Deque
4) ArrayList is better for storing and accessing data.	4) LinkedList is better for manipulating data.
5) an ArrayList is a resizable Array	5) LinkedList implements the doubly linked list of the list interface.

5

Q.No. | Q.No.

प्र. क्र.
Q. No.

Q 7) **StringBuffer** ⇒ Java StringBuffer class is used to create mutable (modified) string objects. The StringBuffer class in Java is the same as String class except it is mutable i.e. it can be changed.

- Java StringBuffer class is thread-safe i.e. multiple threads cannot access it simmultaneously. So it is safe will result in an order

StringBuilder ⇒ Java StringBuilder class is used to create mutable string.
The Java StringBuilder class is same as StringBuffer class except that it is non-synchronised. It is available since JDK 1.5.

10

Scrollable Resultset in JDBC

⇒ A scrollable Resultset is one which allows us to retrieve the data in forword direction as well as backword direction but no updation are allowed. ~~In order to make Non-scrollable Resultset as scrollable Resultset we must use the~~

Q.No. | | | | | | Q.No. | | | | |

प्र. क्र.
Q. No.

Section - C

Q 1)

```
class Actor {
    @Id
    @GeneratedValue(strategy = GenerationType.
                                    IDENTITY)
    private int id;
    @Column(name = "name")
    private String name;

    String String movieName;
    public Actor(){ }
    public Actor(String name, String movieName,
                                    int id)
    {
        this.name = name;
        this.id = id;
        this.movieName = movieName;
    }
    public int getId(){
        return id;
    }
    public void setId(int id)
        { this.id = id; }
    public String setName(){ return Name;}
    public void setName(){
        this.name = name;
    }
```

Q.No. | Q.No.

```
public String getMovieName(){
    return movieName;
}
public void setMovieName(String movieName)
{
    this.movieName = movieName;
}
```

JDBC

```
public class ActorResource{
    private EntityManager entityManager;

    @Post
    public Response addActor(Actor actor){
        entityManager.persist(actor);
        return Response.status(Response.Status.
        CREATED).entity(actor).build();
    }
    @GET
    @Path("/{id}")
    public Response getActor(@PathParam("id") int id)
    {
        Actor actor = entityManager.find(Actor.class,
                find(Actor.class, id);
        if return Response.status(Response.Status
                .NOT_FOUND).build();
    }
```

Q.No. | Q.No.

Q. No.

@PUT

@Path("{id}")

public Response updateActor(@PathParam("id")
int id, Actor updateActor)

{

Actor actor = entityManager.find(
Actor.class, id);

if(actor != null) {

actor.setName(updateActor.getName());

actor.setMovieName(updateActor.getMovie
Name());

entityManager.merge(actor);

return Response.ok(actor).build();

} else {

return Response.status(Response.Status.
NOT_FOUND).build();

}

}

Q 2)

public class MatrixSum {

public static void main(String[] args)

{

int[][] matrix1 = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9} };

Q.No. | Q.No.

```
int[][] matrix2 = {  {9, 8, 7},
                     {6, 5, 4},
                     {3, 2, 1}   };
System.out.println("matrix 1:");
displayMatrix(matrix1);
System.out.println("matrix 2:");
displayMatrix(matrix2);
System.out.println("row sum");
int[] rowSum1 = calculateRowSum(matrix1);
displayArray(rowSum1);
int[] rowSum2 = calculateRowSum2(matrix2);
displayArray(rowSum2);
.
System.out.println("Colum sums:");
int[] colSum1 = calculateColumnSums(matrix1);
displayArray(colSums1);
int[] colSum2 = calculateColumnSums(matrix2);
displayArray(colsum2);
}
private public static void displayMatrix(int[][] matrix)
{  for(int[] row: matrix){
       for(int value: row){
           System.out.println(value + " ");
       }
       System.out.println();
   }
}
```

Q.No. | Q.No.

प्र. क्र.
Q. No.

```
private static int[] calculateColumnSums(int [][]
                                    matrix){
int[] colSums = new int[matrix[0].length];
  for(int j=0; j<matrix[0].length; j++)
  {
    for(int i=0; i<matrix.length; j++)
    {
      colSums[j] += matrix[i][j];
    }
  }
  return colSums;
}
private static void displayArray(int[] array)
  {
    for(int value: array)
    {
      System.out.print(value + "  ");
    }
    System.out.println();
  }
}
```