ASSIGNMENT

1) What is sqi injection, How to resolve the same in java?

A) **SQL injection** is one of the top 10 web application vulnerabilities.

SQL injection means injection/inserting SQL code in a query vis user-inputted data.It can occur in any applications using relational database like Oracle,MySQL,PostgreSQL and SQL server.

There are 4 types of SQL injections:

1. Boolean Based SQL Injection
2. Union based
3. Time-based
4. Error based

To resolve the SQL injection, we can use PreparedStatement instead of Statement to execute the query.For example,Instead of concatenating username and password into the query, we provide them to query via PreparedStatement's setter methods.

2) What is the exception hierarchy?

A) "an event that occurs during the execution of a program that disrupts the normal flow of instructions" is called an exception.it is an unexpected or unwanted event which can occur either at compile-time or run-time in application code. Java exceptions can be of several types and all exception types are organized in a fundamental hierarchy.

The Throwable class is the root class of Java Exception hierarchy inherited by two subclasses

Exception hierarchy classes are two types:

1) checked exception

2) Unchecked exception

Checked exception:

Exceptions that can occur at compile-time are called checked exceptions and The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions.
Eg: IOException, SQLException,InterruptedException

Unchecked exception:

The classes that inherit the RuntimeException are known as unchecked exceptions.
Eg:ArithmeticException,NullPointerException,
ArrayIndexOutOfBoundsException, etc.
Unchecked exceptions are not checked at compile-time, but they are checked at runtime.
EXAMPLE:

```
public class Test{
public static void main(String args[]){
try{
 int data=100/0;
 }catch(ArithmeticException e){
      System.out.println(e);
      }
System.out.println("not divisible.");
  }
}
```

3)How will you decide to choose between interface and abstract?

A) **INTERFACE:**

The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body.

ABSTRACT:

A class which is declared with the abstract keyword is known as an abstract class in Java. It can have abstract and non-abstract methods (method with the body).

An abstract class allows you to create functionality that subclasses can implement or override. An interface only allows you to define functionality, not implement it. And whereas a class can extend only one abstract class, it can take advantage of multiple interfaces.

4)What is the difference between == and equals()?

A) both equals() method and "==" operator in Java are used to compare objects to check equality .

- The main difference between the .equals() method and == operator is that one is a method and the other is the operator.
- We can use "==" operator for reference comparison and .equals() method for content comparison. == checks if both objects point to the same memory location whereas .equals() evaluates to the comparison of values in the objects.

EXAMPLE:

```
public class Test {
public static void main(String[] args)
 {
String s1 = "HELLO";
String s2 = "HELLO";
String s3 =  new String("HELLO");
System.out.println(s1 == s2);
System.out.println(s1 == s3);
```

```
System.out.println(s1.equals(s2));
System.out.println(s1.equals(s3));
    }
 }
```

5) What is the difference between throw and throws?

A) Throw is a keyword which is used to throw an exception explicitly in the program inside a function or inside a block of code. Throws is a keyword used in the method signature used to declare an exception which might get thrown by the function while executing the code.

Throw:

- throw keyword is used throw an exception explicitly in the code, inside the function or the block of code.
- Using throws keyword, we can declare both checked and unchecked exceptions
- The throw keyword is followed by an instance of Exception to be thrown.
- throw is used within the method.
- We are allowed to throw only one exception at a time i.e. we cannot throw multiple exception.

Throws:

- throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code**.**
- the throws keyword can be used to propagate checked exceptions only.
- The throws keyword is followed by class names of Exceptions to be thrown.
- throws is used with the method signature.

- We can declare multiple exceptions using throws keyword that can be thrown by the method. For example, main() throws IOException, SQLException.

EXAMPLE:

```
public class TestThrowAndThrows
        {
Static void method() throws ArithmeticException
{
System.out.println("Inside the method()");
 throw new ArithmeticException("throwing Arithmet
   icException");
    }
 public static void main(String args[])
 {
 try
     {
  method();
        }
   catch(ArithmeticException e)
  {
System.out.println("caught in main() method");
         }
       }
 }
```

6)What is the use of the toString method?

A) toString() is uesd to represent any object as a string, toString() method comes into existence.
The toString() method returns the string representation of the object.

if you print any object, java compiler internally invokes the toString() method on the object. So overriding the toString() method, returns the desired output, it can be the state of an object .

Example:

```
Public class Student{
int rollno;
String name;
String city;
Student(int rollno, String name, String city){
this.rollno=rollno;
this.name=name;
 this.city=city;
}
    public String toString(){
return rollno+" "+name+" "+city;
      }
  public static void main(String[] args){
  Student s1=new Student(1,"Ravi","hyderabad");
   Student s2=new Student(2,"Vijay","andhra");
   System.out.println(s1);
     System.out.println(s2);
    }
   }
```

7)What is immutable in java?

A) **Immutable class** means that once an object is created, we cannot change its content.all the wrapper classes (like Integer, Boolean, Byte, Short) and String class is immutable.

- ❖ The class must be declared as final (So that child classes can't be created)
- ❖ Data members in the class must be declared as private (So that direct access is not allowed)
- ❖ Data members in the class must be declared as final (So that we can't change the value of it after object creation)
- ❖ A parameterized constructor should initialize all the fields performing a deep copy (So that data members can't be modified with object reference)

- Deep Copy of objects should be performed in the getter methods (To return a copy rather than returning the actual object reference)
- No setters (To not have the option to change the value of the instance variable)

EXAMPLE:

```
import java.util.HashMap;
import java.util.Map;
  public final class Student {
  private final String name;
  private final int regNo;
 private final Map<String, String> metadata;

  public Student(String name, int regNo,
                Map<String, String> metadata)
{
        this.name = name;
        this.regNo = regNo;
        Map<String, String> tempMap = new HashMap<>();
        for (Map.Entry<String, String> entry :
              metadata.entrySet()) {
              tempMap.put(entry.getKey(), entry.getValue());
        }
        this.metadata = tempMap;
}

public String getName() { return name; }

public int getRegNo() { return regNo; }

public Map<String, String> getMetadata()
{
        Map<String, String> tempMap = new HashMap<>();
        for (Map.Entry<String, String> entry :
```

```
                this.metadata.entrySet()) {
                tempMap.put(entry.getKey(), entry.getValue());
            }
            return tempMap;
        }
}

// Driver class
class Test {
        public static void main(String[] args)
        {
            Map<String, String> map = new HashMap<>();
            map.put("1", "first");
            map.put("2", "second");
            Student s = new Student("ABC", 101, map);
            System.out.println(s.getName());
            System.out.println(s.getRegNo());
            System.out.println(s.getMetadata());



            map.put("3", "third");
            System.out.println(s.getMetadata());
            s.getMetadata().put("4", "fourth");
            System.out.println(s.getMetadata());
        }
}
```

8) What fails fast in collections mean, how can we resolve it?

A) Using iterations we can traverse over the collections objects. The iterators can be either **fail-safe** or **fail-fast**.

Fail-fast iterator throw an exception if the collection is modified while iterating it.

Fail-safe iterators means they will not throw any exception even if the collection is modified while iterating it.

EXAMPLE:

```
ArrayList<Integer> integers = new ArrayList<>();
integers.add(1);
integers.add(2);
integers.add(3);
Iterator<Integer> itr = integers.iterator();
while (itr.hasNext()) {
Integer a = itr.next();
integers.remove(a);
   }
```

9)What is the benefit of string tokenizer ?

A) The **string tokenizer** class allows an application to break a string into tokens. The tokenization method is much simpler than the one used by the StreamTokenizer class. The StringTokenizer methods do not distinguish among identifiers, numbers, and quoted strings, nor do they recognize and skip comments.

10)How are String, String Buffer and String Builder different from each other?

A)The differences between String, StringBuffer, and StringBuilder are based on the following two parameters:

- Mutability
- Performance

Mutability:

when we compare the above terms, on the factor Mutability. the Strings are immutable while StringBuffer and StringBuilder are mutable. So, Strings cannot be changed when

you use the String class; whereas Strings can change if you use the StringBuffer and StringBuilder class.

PERFORMANCE:

StringBuilder is faster than StringBuffer as it offers no synchronization. This is because no extra overhead needs to be added to the system and also does not slows down the processing.