**1. Explain the difference between primitive and reference data types.**

* **primitive data type always has a value, it can never be null but reference type can be null**, which denotes the absence of value. primitive types store actual values but reference type stores handle to object in the heap.

**2. Define the scope of a variable (hint: local and global variable)**

* Global  variables are**declared outside of a function for accessibility throughout the program,** while local variables are stored within a function using var for use only within that function’s scope.

**3. is initialization of variables required.**

* Initializing a variable can**prevent bugs that is when it happens that** variable is a reference type, initializing it can prevent null reference errors down the line.

**4.Differentiate between static, instance and local variables.**

* local Variables declared inside a method and only exist while that method is invoked. Static variables are similar to instance variables except that they belong to the actual Class object rather than a specific instance of the class, and hence the SAME variable can be accessed from all instances of the class.

**5.Differentiate between widening and narrowing casting in java.**

* Narrowing casting involves the conversion of a higher data type into a lower one also called explicit conversion or casting up while widening casting is conversion of smaller data type into larger size that’s byte to short among others.

**6.The following table shows data type, its size, default value and the range. Filling in the missing values.**

|  |  |  |  |
| --- | --- | --- | --- |
| types | size (In byte) | Ddefault | Range |
| Boolean | 1 bit | false | true, false |
| Char | 2 | ‘\u0000’ | ‘\0000’ to ‘\ffff’ |
| Byte | 2^8 | 0 | -27 to +27-1 |
| Short | 2 bytes | 0 | -215 to +215-1 |
| Int | 4 | 0L | -231 to +231-1 |
| Long | 8 | 0L | - 2^63 |
| Float | 4 | 00.0f | 3.4E-38 to 3.4E+38 |
| Double | 8 | 0.0d | -1.8E+308 to +1.8E+308 |

**7.Explain the importance of using Java packages**

* **Java package** is used to categorize the classes and interfaces so that they can be easily maintained.
* Java **package** provides access protection.
* Java package removes naming collision.

8**.Explain three controls used when creating GUI applications in Java language.**

* **Label** is a component that is used to define a simple text on the screen
* **The Radio Button** is used to provide various options to the user. The user can only choose one option among all.
* **Text Field** is basically used to get the input from the user.

**9.Explain the difference between containers and components as used in Java.**

* The **class Component** is the abstract base class for the non-menu user-interface controls of AWT. A **component** represents an object with graphical representation while **the class** **Container** is the superclass for the **containers** of AWT. The**container** object can contain other AWT components.

**10.Write a Java program to reverse an array having five items of type int**.

public class ReverseArray {

//function that reverses an array and stores it in another array.

static void reverse(int a [],int n)

{

int [] b=new int[n];

int j= n;

for (int i=0;i<n; i++){

b[j-1]=a[i];

j=j-1;

}

// printing the reverse array

System.out.println("Reversed array is\n");

for (int k=0;k<n;k++){

System.ousst.println(b[k]);

}

}

public static void main (String [] args) {

int [] arr={5,10,15,20,25};

reverse (arr, arr. length);

}

}

**11.Programs written for a graphical user interface have to deal with “events.”**

**Explain what is meant by the term event.**

**Give at least two different examples of events, and discuss how a program might**

**respond to those events.**

Define the term event; its the change in the state of an object.

12**.Explain the difference between the following terms as used in Java programming**.

**Polymorphism and encapsulation**

* Encapsulation can hide some of the private details of a class from other objects, while polymorphism allows us to use a common operation in different ways.

**method overloading and method overriding**

* Method overloading is used to increase the readability of a program while method overriding is used to provide specific implementation of method that is already provided by its super class

**class and interface**

* Class’s object can be created while interface object can not be created
* Class doesn’t support multiple inheritance while interface supports multiple inheritance.

**inheritance and polymorphism**

* Inheritance supports the concept of reusability and reduces the code length in object-oriented programming while polymorphism allow the object to decide which form of the function is to implemented at compile time as well as run time.

**13.sing examples, explain the two possible ways of implementing polymorphism. Show your code in java.**

* Polymorphism can be performed in two different ways that is through**; method overriding** and **method overloading.**
* **Method overloading** a concept of Java in which we can create multiple methods of the same name in the same class, and all methods work in different ways.
* **Method overriding** occurs when the method signature is the same in the superclass and the child

class class Language {

public void displayInfo() {

System.out.println("Common English Language");

}

}

class Java extends Language {

@Override

public void displayInfo() {

System.out.println("Java Programming Language");

}

}

class Main {

public static void main(String[] args) {

// create an object of Java class

Java j1 = new Java();

j1.displayInfo();

// create an object of Language class Language

l1 = new Language();

l1.displayInfo();

}

}

**Method Overloading**; occurs when two or more methods in the same class have the same name but different parameters.

Classpattern**{**

// method without parameter

public void display(){

for (int i = 0; i < 10; i++) {

System.out.print("\*");

}

}

// method with single parameter

public void display(char symbol) {

for (int i = 0; i < 10; i++) {

System.out.print(symbol);

}

}

}

class Main { public static void main(String[] args) {

Pattern d1 = new Pattern();

// call method without any argument

d1.display();

System.out.println("\n");

// call method with a single argument

d1.display('#');

}

}

1. **With relevant examples, explain the following concepts as used in Java programming.**

**a. Mutable classes**.

Explain what is meant by mutable class

* A mutable class is **one that can change its internal state after it is created**

Write a program that implements the concept of mutable class

1. public class JtpExample {

2. private String s;

3. JtpExample(String s) {

4. this.s = s;

5. }

6. public String getName() {

7. return s;

8. }

9. public void setName(String coursename) {

10. this.s = coursename;

11. }

12. public static void main(String[] args) {

13. JtpExample obj = new JtpExample("JavaTpoint");

14. System.out.println(obj.getName());

15. // Here, we can update the name using the setName method.

16. obj.setName("Java Training");

17. System.out.println(obj.getName());

18. }

19. }

**Explain what is meant by immutable class**

b**. Immutable classes.**

* immutable classes define objects which, once created, never change their value

**Write a program that implements the concept of immutable class**

1. public class JtpExample1 {

2. private final String s;

3. JtpExample1(final String s) {

4. this.s = s;

5. }

6. public final String getName() {

7. return s;

8. }

9. public static void main(String[] args) {

10. JtpExample obj = new JtpExample("Core Java Training");

11. System.out.println(obj.getName());

12. }

13s. }

c. **Explain the situations where mutable classes are more preferable than immutable classes when writing a Java program.**

* when you have large objects, creating a single copy becomes expensive. So, creating multiple copies of an object becomes less efficient and slows down your computer's operation. In such a situation, a mutable object is better than an immutable object.

2. **Explain what a String buffer class is as used in Java**

* String Buffer is a peer class of String that provides much of the functionality of strings.

**the syntax of creating an object of String Buffer class**

public class String buffer {

public static void main (String [] args) {

String Buffer sb = new String Buffer("study");

System.out.println(sb);

// modifying object

sb.append("tonight");

System.out.println(sb) ; // Output: study tonight

}

**}**

**Explain the methods in the String Buffer class**

**Insert () method**

The insert () method inserts the given string with this string at the given position.

**Replace () method**

The replace () method replaces the given string from the specified begin Index and endIndex-1.

**Delete () method**

The delete () method of String Buffer class deletes the string from the specified begin Index to endIndex-1

The reverse () method of StringBuilder class reverses the current string.

The capacity () method of String Buffer class returns the current capacity of the buffer.

2b. **Write the output of the following program**.

class Myoutput

1. {
2. public static void main(String args[])
3. {
4. String ast = "hello i love java";
5. System.out.println(ast.indexOf('e')+" "+ast.indexOf('ast')+" "+ast.lastIndexOf('l')+" "+ast .lastIndexOf('v'));
6. }
7. }

**c. Explain your answer in (2b) above**

The program code doesn’t provide the out its void since all the string the line four is not enclosed thus creating error result.

**e. With explanation, write the output of the following program.**

class Myoutput

1. {
2. public static void main(String args[])
3. {
4. StringBuffer bfobj = new StringBuffer("Jambo");
5. StringBuffer bfobj1 = new StringBuffer(" Kenya");
6. c.append(bfobj1);
7. System.out.println(bfobj);
8. }
9. }