**JAVA INTERVIEW QUESTIONS – MISCLAENIOUS**

**JAVA – 8**

**Topic – wrapper class**

1. **What are wrapper classes in java?**

* Wrapper classes are those classes provided by java to make objects of primitive data types. As java is an object-oriented programming language so sometimes we need objects rather than primitive types.

Using wrapper class we can do this.

1. **What is autoboxing & unboxing?**

* In java from java SE 5.0.0 java provides automatic conversion from primitive types into the object types it is known as **autoboxing.**
* **Example --> Byte to Byte, char to Character(wrapper class) int to Integer(wrapper c…), etc**
* From object into primitive types, the conversion is known as **unboxing.**

1. **How we can perform autoboxing and unboxing in java?**

* In java, we can perform explicit & implicit autoboxing and unboxing.
* Autoboxing 🡺
  + Explicit & Implicit 🡺

//Autoboxing example of int to Integer

* 1. **public** **class** WrapperExample1{
  2. **public** **static** **void** main(String args[]){
  3. //Converting int into Integer
  4. **int** a=20;
  5. Integer i=Integer.valueOf(a);//converting int into Integer explicitly
  6. Integer j=a;//autoboxing, now compiler will write Integer.valueOf(a) internally

* 1. System.out.println(a+" "+i+" "+j);

}}

* Unboxing 🡺
  + Explicit & Implicit 🡺

Integer a=**new** Integer(3);

**int** i=a.intValue();//converting Integer to int explicitly  **int** j=a;//unboxing, now compiler will write a.intValue() internally

1. **Why java is not a fully object-oriented programming language?**

* Java is not considered as fully object-oriented because of two specific reasons-->
  + - **Static keyword:**

we can access any data member of a class if that is declared with static we can access without creating any instance of that class.

* + - **Primitive types:**

In java we can use primitive data like int, float, char, etc those are not object that also violet the object-oriented concepts as we can directly use them.

Topic – objects & Access Modifiers

1. **How can we pass multiple arguments of same type to any method in java means when the number of arguments are specified?**

* We can do this using three dot operators after the type declaration inside the method signature it will treat the argument as a array nothing else.
  + - * + {Public static void main(String args [ ] )
        + {
        + Void print(5 , 6 , 6 , 7 , 8 , 0 , 10 );
        + }
        + Void print(int…a)//a is considered as a integer arrays
        + {
        + For(int I : a)
        + System.out.println(i);
        + } }

1. **Object vs class In java**

|  |  |  |
| --- | --- | --- |
|  | **Object** | **Class** |
| 1) | Object is an **instance** of a class. | Class is a **blueprint or template** from which objects are created. |
| 2) | Object is a **real world entity** such as pen, laptop, mobile, bed, keyboard, mouse, chair etc. | Class is a **group of similar objects**. |
| 3) | Object is a **physical** entity. | Class is a **logical** entity. |
| 4) | Object is created through **new keyword** mainly e.g. Student s1=new Student(); | Class is declared using **class keyword** e.g. class Student{} |
| 5) | Object is created **many times** as per requirement. | Class is declared **once**. |
| 6) | Object **allocates memory when it is created**. | Class **doesn't allocated memory when it is created**. |
| 7) | There are **many ways to create object** in java such as new keyword, newInstance() method, clone() method, factory method and deserialization. | There is only **one way to define class** in java using class keyword. |

1. **What are the different type of modifiers in java?**

* In java there are two type of modifiers 🡺
  + - Access Modifiers
    - Non-Access Modifiers (Abstract keyword)

1. **What are the use of Access Modifiers?**

* We can use access modifiers to maintain accessibility of fields, methods, classes, data members. Using access specifiers we can able to set different accessibility for different fields for better security.
  + - There are four type of access modifiers in java
      * **Private**
      * **Default**
      * **Protected**
      * **Public**

1. **Explain Different Access Modifiers and their accessibility**

* **Private:**
  + - When any field, methods or data members are declared with private keyword then those fields are not accessible from outside the class.
    - The scope of these type of members are only limited within the same class itself.
* **Default:**
  + - When any field, methods are declared with default then the privileges are increased than the private. Because fields declared with default can be accessible from same package but from the different packages those members can not be accessible.so default is less restricted than private and in java default is by default means it’s a by default access modifier.
* **Protected:**
  + - Protected members, fields, methods are more less restricted and can be accessible from anywhere in the same package but can be accessible from another package but from the subclasses only. Rather than that it’s not possible.
* **Public:**
  + - Public access modifier if that is declared with any member then it can be accessible by any class from anywhere outside of same package, inside from the same package.

1. **Differences of Access specifiers in a diagram**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **within class** | **within package** | **outside package by subclass only** | **outside package** |
| **Private** | Y | N | N | N |
| **Default** | Y | Y | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |

1. **What is singleton class in java?**

* When constructor of a class is declared with private at that moment only one object can be created for a class.

We can not create object of singleton class using constructors but we can do it through getInstance() method.