ISE – 1

Q.1.

a. Under what circumstances does the "variable might not have been initialized" compilation error occur?

Ans: The “variable might not have been initialized” compilation error occurs in Java when you attempt to use a variable that has not been assigned a value yet.

Using a local variable without initializing it:

When we declare a variable but we fail to assign a value to it and use it without assigning a value.

Example –

```
int x;

System.out.println(x);
```

Using an instance or class variable without initializing it:

Similar to local variables, instance and class variables must be initialized before they are used.

Example –

```
public class MyClass {
   int y;
   public void printY() {
      System.out.println(y);
   }
}
```

b. Assuming the given registration number "2024bit025" is a valid registration number. How can one determine if it belongs to the IT department, assuming the code for the IT department is 'bit'? Provide an alternative method from the String class for checking this without using the charAt() method and explain how to utilize the alternative method in short.

Ans: To determine whether “2024bit025” belongs IT department or not we can use substring() method.

There are two types of substring method - String String.substring(int beginIndex) and

String String.substring(int beginIndex, int endIndex)

Here we have to check for the ‘bit’ code so we have to use second method which is –

String String.substring(int beginIndex, int endIndex) .

This method return a string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index (endIndex-1) .

Thus the length of the substring is endIndex – beginIndex

beginIndex is inclusive and endIndex is exclusive.

It gives IndexOutOfBoundException Error if the beginIndex is negative or endIndex is larger than the length of this String object or beginIndex is larger than endIndex.

Implementation –

```
String registrationNo = "2024bit025";
String itDeptCode = "bit";
if(registrationNo.substring(4,7).equalsIgnoreCase(itDeptCode)) {
        System.out.println("Registration No. belongs to IT Department");
} else {
        System.out.println("Registration No. does not belong to IT Department");
}
```

Q.2.

Carefully examine the following code snippet before responding to the questions.

```
class ISE {
        Integer String;
        static int year = 2024;
        public static void main(String[] args) {
                ISE jp = new ISE();
                System.out.println(jp.String);// line6
                short bit = (byte) year;
                System.out.println(bit); // line 8
        }
}
```

a. **What is the significance of the static keyword?**

Ans: The 'static' keyword in java is used to declare a member that belongs to the class itself rather than to instance of the class.

In the given code snippet – the variable ‘year’ is declare as static int year = 2024;

Which means it belongs to the class ISE and can be accessed without creating an instance or object of the class ISE. Also the main method is also declared as static so it can be invoked without an instance of the class.

b. Determine the output of line 6 and line & in the following code snippet. Clearly indicate errors, if any, or output printed with proper justification. Without justification, no marks will be awarded.

Ans:
Output of line 6: null

Explanation: Since ‘String’ is an object reference variable and not initialized, its default value will be ‘null’. Hence, it will print ‘null’.

Output of line 8: -24

Explanation: The year variable is declared as ‘static int year = 2024’, and it’s not within the range of byte. Hence, when it is cast to byte, it will overflow. In java, integer overflow wraps around, so the result will be -24. Size of byte is (2^8) that is 256.

In jshell –

jshell> static int year = 2024;

year ==> 2024

jshell> short bit = (byte)year;

bit ==> -24

OR

jshell> 2024 % 256

$5 ==> 232

jshell> 232 - 256

$6 ==> -24