1. Polymorphism in java is the ability of an object to take many forms. It allows us to perform the same action in many different ways. It occurs when we have many classes that are related to each other by inheritance.
2. overloading a method means the methods have the same name but different parse in parameters.
3. overriding a method means the subclass overrides the same method with the same name from the parent class.
4. the final in this method means the Car object can not be reassigned after being parsed into this method. However, we can still change the attributes that are defined inside this object.
5. Yes, we can call the aCar.setColor(red) because we can still change the attribute inside this Car object.
6. No, we can not declare a static variable inside a method, because the static means that's a variable/method of a class, it belongs to the whole class but not one of its certain objects. So static keyword can be used only in the class scope.
7. interface vs abstract class:
   Type of methods: Interface can have only abstract methods. An abstract class can have abstract and non-abstract methods. From Java 8, it can have default and static methods also.
   Final Variables: Variables declared in a Java interface are by default final. An abstract class may contain non-final variables.
   Type of variables: Abstract class can have final, non-final, static and non-static variables. The interface has only static and final variables.
   Implementation: Abstract class can provide the implementation of the interface. Interface can't provide the implementation of an abstract class.
   Inheritance vs Abstraction: A Java interface can be implemented using the keyword "implements" and an abstract class can be extended using the keyword "extends".
   Multiple implementations: An interface can extend another Java interface only, an abstract class can extend another Java class and implement multiple Java interfaces.
   Accessibility of Data Members: Members of a Java interface are public by default.
   A Java abstract class can have class members like private, protected, etc.

8. Yes, we can declare an abstract class with no abstract methods in Java. An abstract class means that hiding the implementation and showing the function definition to the user. An abstract class having both abstract methods and non-abstract methods

9. The purpose of an abstract class is to function as a base for subclasses.

10. A native method, Simply put, this is a non-access modifier that is used to access methods implemented in a language other than Java like C/C++. For example:
    In java following:

```
public native int intMethod(int i);
public static void main(String[] args) {
   System.loadLibrary("Main");
   System.out.println(new Main().intMethod(2));
}
```

```
In programing C:
#include <jni.h>
#include "Main.h"
JNIEXPORT jint JNICALL Java_Main_intMethod(
JNIEnv *env, jobject obj, jint i) {
```

11. It is an empty interface (no field or methods). Examples of marker interface are Serializable, Cloneable and Remote interface. All these interfaces are empty interfaces.

12. override equals can help redefined for the object to compare each other with our purpose. override the hashCode can help any collection based class functioning properly in conjunction as its purpose for example HashMap, hashSet, hashTable.

13. the int is primitive type type and it has less flexibility, however the Integer is a wrapper class and it has more flexibility to store, convert and manipulate the data.

14. Serialization is the conversion of the state of an object into a byte stream; deserialization does the opposite. Stated differently, serialization is the conversion of a Java object into a static stream (sequence) of bytes, which we can then save to a database or transfer over a network.

15. Check the code in project path src/main/java/day4/Assignment3.java

16. Check the code in project path src/main/java/day4/Assignment3.java