**Core Java**

**1.** **Diff b/w String and String Buffer String Builder**

**● String objects are immutable - non changeable- once u created object , we can’t change existing objects**

**● String buffer objects are mutable - changeable**

**Eg 1: string s= new String(“Bhuva”);**

**s.concat(“vajra”);**

**s.o.p(s); // Bhuva**

**- Here string is used. so , concat not change existing ‘s’ value. Also it is not assigned to any variable. So o/p is Bhuva only**

**- If content is fixed and won't change frequently then go for String**

**Eg 2: StringBuffer sb= new StringBuffer(“durga”);**

**sb.append(“software”);**

**s.o.p(s); //durga software**

**- Here software appended with existing sb object . which means it is changeable**

**- Hence it is called mutability**

**- If content is not fixed and keep on changing then go for string buffer or string builder**

**- Content is not fixed but thread safety is required. Only on thread should access object at a time then go for string buffer. In string buffer every method is synchronized. And performance wise not good. Bcz other threads needs to wait until the current thread completes.**

**- Don't want thread safety, multiple threads can access at a time, and content is mutable then go for string builder. Performance wise recomanded bcz every method is non synchronized method**

**2. What is static**

**● In java static keyword mainly used for memory management by sharing the variable or method of same class**

**● Static is used describe the constant variable or method same for every instance of class**

**● Static member can be variable,methods,blocks,nested classes**

**● Can access static member without object reference**

**Eg programs :**[**https://www.edureka.co/blog/static-keyword-in-java/**](https://www.edureka.co/blog/static-keyword-in-java/)

**a.** **Static block**

**● Static block can be executed only once when the class is first loaded**

**● Static block can be executed before main method**

**b.** **Static variable**

**● Static variable are global variable and can be created at the class level**

* **Single copy of a static variable can be created and divided among all the objects at class level.**

**c.** **Static methods**

**● Static method can directly call static data, static methods only eg: main method**

**● Inside main method, if we want to call any method we need to create object and then need to call**

**● But for static methods we don't need to create objects, we can directly call method/variable.**

**d.** **Static classes**

**● Class can be made static only if it is nested class**

**● Nested static class does not require any reference of outer class**

**3.What is final**

**● Final keyword can be variable, method,class**

**Eg:** [**https://www.javatpoint.com/final-keyword**](https://www.javatpoint.com/final-keyword)

**a.** **Final variable**

**● Final variable value is can not be changed.it become constant**

**● Final variable without value is blank or uninitialized final variable**

**● fInal variable can be initialized at constructor only**

**● If final variable is static then it can be initialized in static block only**

**b.** **Final method - final method cannot be overridden. But can be inherited**

**c.** **Final class cannot be extend/inherited**

**4. Serialization:**

**● The process of converting object from java supported form into network or file supported form is known as serialization**

**● Using fileoutputstream and object output stream classes we can achieve this**

**5. Synchronization**

**● In multi threading program sometime multiple thread may access the same resource at the same time, finally it produce the erroneous output**

**● So by using synchronization we can ensure that only one thread can access the resource at a time**

**● It can be achieved by using synchronized block**

**6.what is Exception Handling**

[**https://beginnersbook.com/2013/04/java-exception-handling/**](https://beginnersbook.com/2013/04/java-exception-handling/)

**● An exception is an unwanted event that interrupt the normal flow of the program**

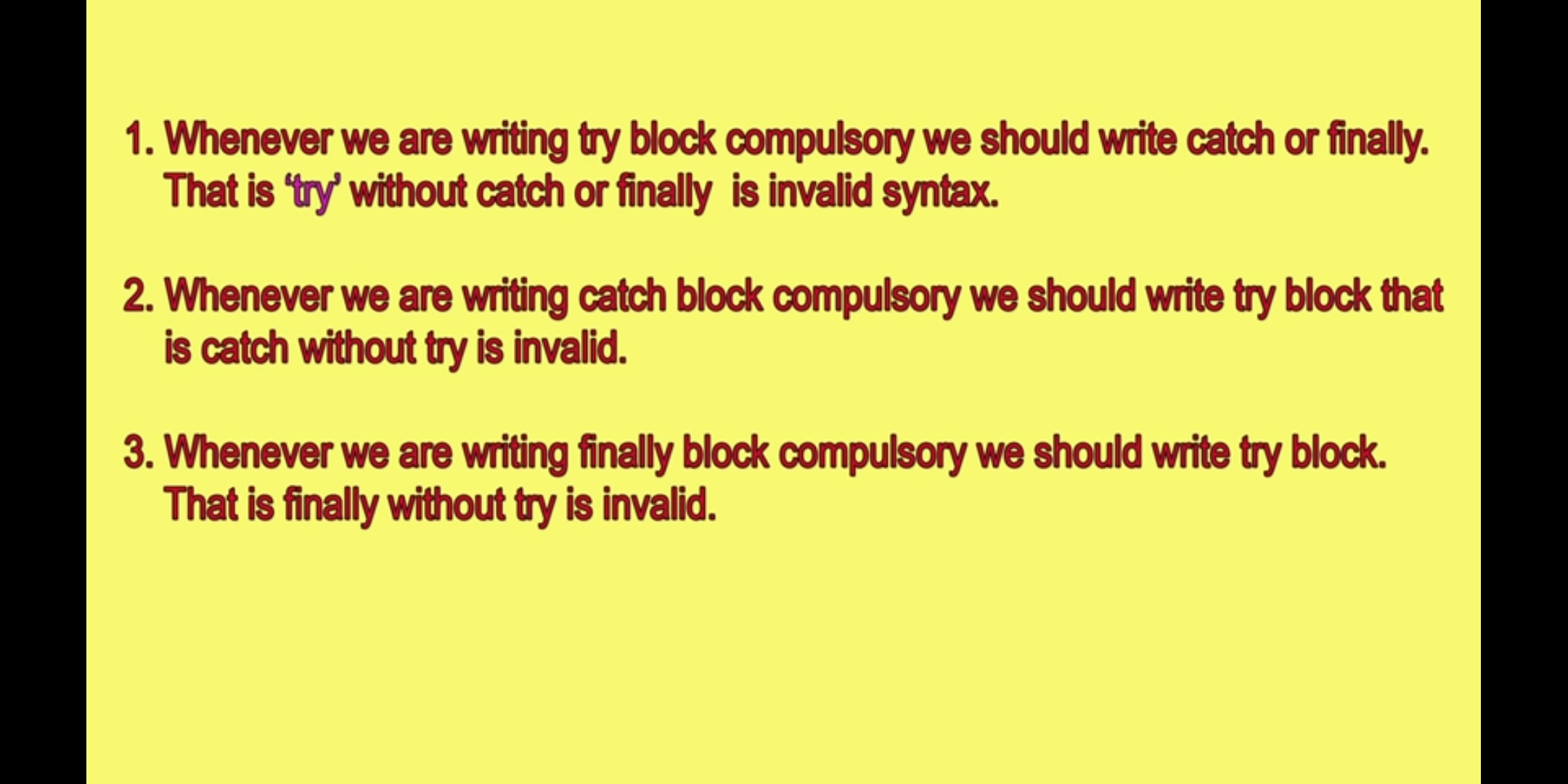
**● When an exception occurs program gets abnormally terminated and system generated error messages throws**

**● By handling this exception we can provide meaningful error message which user can understand**

**● example: Opening a non-existing file in your program, Network connection problem, bad input data- divided by zero**

**● Advantage: flow of program does not break- normal termination**

**Different types of try catch combination,**

****

[**https://drive.google.com/drive/folders/1HpHsEY7y6PlzhlJg8NneAW6tActY9\_2F**](https://drive.google.com/drive/folders/1HpHsEY7y6PlzhlJg8NneAW6tActY9_2F)

**7. Error vs Exception:**

**Throwable class is the root for java exception hierarchy**

**a.** **Error - errors can not be handled.if something goes wrong application will crash. error s are not occured due to our program. These are occurred due to lack of system resources.**

**Errors are non recoverable**

**eg:Out of memory error**

**b.** **Exception - events that occur in the code. Programmer can handle such conditions and take actions.**

**Most of the cases Exceptions are occurred by our programm only and those are recoverable.**

**Eg: file not found exception**

**8. Type of Exceptions**

**1.** **Checked exceptions**

**● All exceptions other than Runtime Exceptions are known as Checked exceptions as the compiler checks them during compilation to see whether the programmer has handled them or not.**

**● If these exceptions are not handled/declared in the program, you will get compilation error.**

**● For example, SQLException, IOException, ClassNotFoundException etc.**

1. **Partial Checked exception - Exception(some child classes are un checked), Throwable(Error is un checked exception)**
2. **Fully checked exception - IO Exception,**

**2.** **Unchecked exceptions**

**● Runtime Exceptions are also known as Unchecked Exceptions.**

**● These exceptions are not checked at compile-time so compiler does not check whether the programmer has handled them or not but it’s the responsibility of the programmer to handle these exceptions and provide a safe exit.**

**● For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc.**

**9. Multi threading**

**Thread: A thread is a light-weight ,independent and smallest part of a process that can run concurrently with the other parts(other threads) of the same process**

* **All threads of a process share the common memory.**
* **It is save cpu time**
* **The process of executing multiple threads simultaneously is known as multithreading.**

**A thread can be in one of the following states:**

**1.** **NEW – A thread that has not yet started is in this state.**

**2.** **RUNNABLE – A thread executing in the Java virtual machine is in this state.**

**3.** **BLOCKED – A thread that is blocked waiting for a monitor lock is in this state.**

**4.** **WAITING – A thread that is waiting indefinitely for another thread to perform a particular action is in this state.**

**5.** **TIMED\_WAITING – A thread that is waiting for another thread to perform an action for up to a specified waiting time is in this state.**

**6.** **TERMINATED – A thread that has exited is in this state.**

**A thread can be in only one state at a given point in time.**

**10. Difference between final finally,finalize**

**1.final**

**· Final is modified applicable for classes, methods,variable.**

**· If class declared as final we can't extend the class, i.e we cant create child class for that class**

**· If a method declared as final we can't override that method in the child class**

**· If a variable declared as final then it will become constant and we cant perform re assignment for that variable**

**2.finally – is a block always associated with try catch to maintain the cleanup code like re allocation task. Finally block always executed whether the exception occurred or not, if we call system.exit() then only it will not executed**

**3.finalize method - it is invoked by garbage collector before destruction of object to perform clean up activities**

**11. diff b/w == and equals() method**

**String s1= new string(“durga”);**

**String s2= new string(“durga”);**

**== operator used to compare address or reference**

**s.o.p(s1==s2);**

**equals() method used to compare content of the object**

**s.o.p(s1.equals(s2))**

**12.Access Specifiers vs Access Modifiers**

**In java there is no Access specifiers all are by default access modifiers**

1. **· public**
2. **· private**
3. **· protected**
4. **· default**
5. **· final**
6. **· static**
7. **· synchronized**
8. **· abstract**
9. **· native**
10. **· Strictfp**
11. **· Transient**
12. **· Volatile**

**For top level classes below modifiers only allowable**

**1.** **Public**

**2.** **Datault**

**3.** **Final**

**4.** **Abstract**

**5.** **Strictfp**

**13. when we go for Interface , abstract, concrete class**

**Interface – if we don't have any idea on implementation , only know about requirements then we can go for interface. Eg:servlet**

**Abstract – if we can able to provide partial implementation only.eg: generic servlet.http servlet**

**Concrete class- if we want provide full implementation, eg: own servlet class**

**14. system.out.println();**

**System – class present in java.lang package**

**Out is static variable present in system class of printstrem type.– (static PrintStream out)**

**Println() - it is a method present in PrintStream class**

**15. Difference between Interfaces and abstract class**

**1.Interface**

**· It will talk about only requirement nothing about implementation**

**· Every method is public and abstract even we are not declaring. So it is called 100% pure abstract class**

**· We can't declare interface methods with private,protocted. Bcz already it is public by default. We can't declare with final,static,synchronized,native,strict fp bcz already it is abstract by default**

**· By default every variable public,static,final**

**· Serialization concepts applicable for objects. Transient keyword applicable for serialization. Since we can't create an object for the interface, transient is NA here. Also all the variable are final so we can’t declare with private,protected,transient,volatile**

**· In interface we need to do initialization for variable at the time of declaration**

**· We cant declare any static or instance block**

**· Constructor is not applicable**

**2. Abstract class**

**· It will talk about partial implementation**

**· In addition to abstract methods we can have concrete methods. Need not to be public and abstract all the methods**

**· There are no restrictions on method modifiers**

**· No restriction on variable**

**· No restrictions on variable modifiers**

**· We can declare variable w/o initialization**

**· We can create instance or static block**

**· We can take instance variable and initialize in constructor**

**16. Diff between overriding and overloading**

**Overloading**

**1. method name must be same**

**2. argument type must be different , at least order**

**3. method signature( method name and argument) must be different**

**4.no restriction on return type**

**5. private,static,final method overloaded**

**6. no restriction for access modifiers**

**Overriding**

**1.** **Method name must be same**

**2.** **Argument types must be same including order**

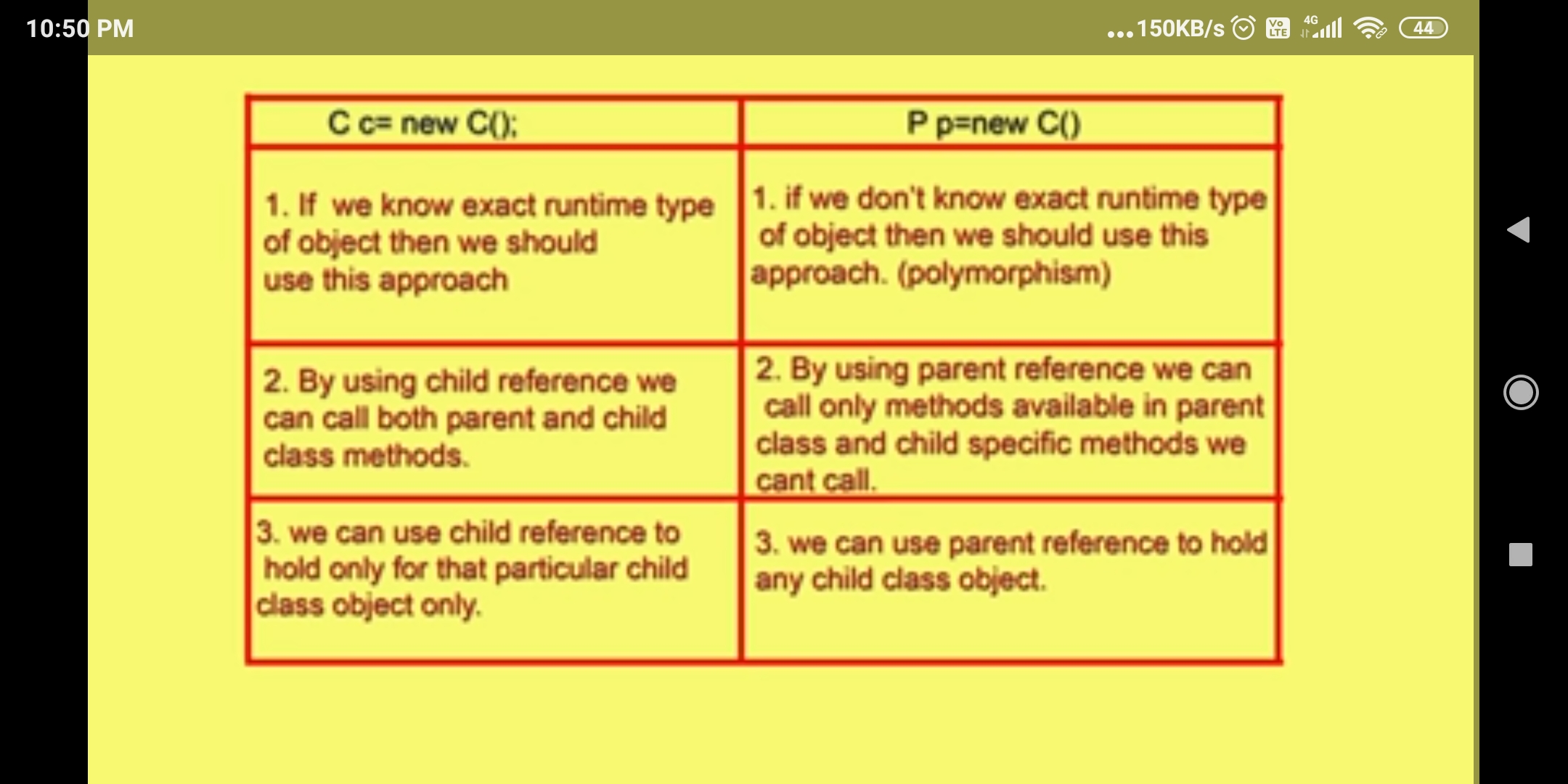
**3.** **Method signature must be same**

**4.** **Covariant return types are allowed, child class return type may be parent class child type or return type should be same**

**5.** **Overriding not applicable for private, static(method hiding) , final**

**6.** **Same or high level scope of modifiers allowed.eg:parent default , child – default,protected,public**

**17) diff b/w C c=new C() and { p =new C()**

****

**18) Polymorphism**

* **Same reference with multiple forms (child)is the concept of polymorphism**
* **Usage of parent reference to hold child object is know as polymorphism**
* **Eg: List l =new AL(), List l =new LL(), List l =new vector()**

**19) Generics**

* **The purpose of Generics to provide type safety and to resolve type casting problem**

**Type Safety**

1. **Type safety means it should allow only the particular type . by default Array is type safety.**
2. **Eg:String[] s. Here we can insert string value only, if we try to insert any other value it will throw CE.**
3. **Collection objects are not type safety by default. if our requirement is only add the string object if we added integer by mistake. CE won’t throw error.**

**Type Casting:**

1. **While retrieving array elements there is no need to provide type casting.**
2. **While retrieving the elements from AL we should perform typecasting.**
3. **Eg: String name= l.get(0); // CE throws . bcz l.get(0) might be any type. So we can store any type into string type.**

* **To resolve above 2 problems generics introduced in 1.5v**
* **Polymorphism concept applicable only for base type not for parameter type**
* **Eg: AL<Object> l = nw AL<String>(); // throws CE : Incompatible types found arraylist of string ,require AL of objects.**
* **We can’t use primitive data type as parameter type and it must be object type(any class or interface name)**

**20) Difference b/w normal array list and generic array list**

**21) diff b/w comparable and comparator?**

|  |  |
| --- | --- |
| **Comparable** | **Comaprator** |
| **It is meant for default natural sorting order** | **Customized sorting order** |
| **Present in java.lang package** | **java.utill package** |
| **It contains only one method compareTo()** | **I/F contains 2 methods. 1.compare(), 2)equals()** |
| **All wrapper classes and string classes are implements comparable** | **2 classes implements,**  **1.collator, 2.RuleBasedCollator** |

**22.this keyword**

**23.Cloning**

**24.Shallow Compare vs deep compare**

**26.**