Assignment 5 :

(1) Why are strings not preferred for storing passwords? Why use character sequence or char arrays instead?

→ Since strings are immutable in Java, if you store the password as plain text it will be available in memory until the Garbage collector clears it, & since string is used in the string pool for reusability there is a pretty high chance that it will remain in memory for long duration, which pose a security threat.

↳ since strings are immutable then is no way the contents of strings can be changed bec/ any change will ~~change~~ produce a new string, while if you use a char[] you can still set all the elements as blank or zero. So storing a password in a character array clearly Mitigates the security risk of stealing a password.

Eg: String strPassword = "Unknown";

char[] charPassword = new char[] {'U', 'N', 'K', 'N', 'O', 'W', 'N'};

S.O.P ("String password:" + strPassword);

S.O.P ("Character Password" + charPassword);

O/p: String Password : unknown

Character Password : [C@110b053

3) Strings are immutable. Explain.

↳ In Java, String objects are immutable, immutable simply means unmodifiable / unchangeable. Once String object is created its data or state can't be changed but a new String object is created. Because Java uses the concept of "String literals".

Ex: String s1 = "Java";

s1.concat ("ruler");

S.O.P ("s1 refers to:" + s1);

O/p: s1 refers to Java

Here first line is straightforward, in next line, the VM creates another new String "Java ruler", but nothing refers to it. So it is instantly lost.

Note: JVM sets aside a special area of memory called "String Constant Pool", to reduce the cause of redundancy of memory & to make Java more efficient

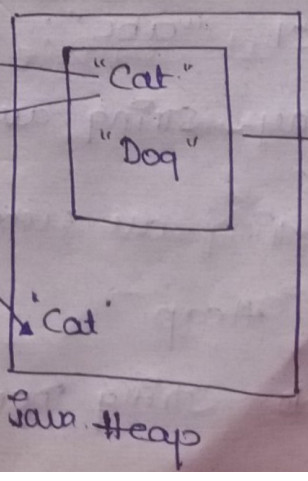
③ What is difference b/w String, StringBuffer & StringBuilder.

Parameter	String	StringBuffer	StringBuilder
① storage	stringpool	Heap	Heap.
② mutability	immutable	mutable	mutable
③ Thread safe	not used in a threaded environment	used in a multithreaded environment	used in a single threaded environment
④ Performance	slow	slower than StringBuilder but faster than String.	faster than StringBuffer
⑤ Syntax:	String s = "Hello"; String s = new String ("Hello");	StringBuffer s = new StringBuffer ("Hello");	StringBuilder s = new StringBuilder ("Hello");

④ Explain the concept of String Pool/intern. Why it is required?

→ String pool in Java is a pool of strings stored in Java heap memory. (String is a special class in Java)

```
String s1 = "cat";
String s2 = "cat";
String s3 = new String("cat");
s1 == s2; // true;
s1 == s3; // false
```



→ String pool helps in saving a lot of space for Java runtime although it takes more time to create the string.

→ When we use double quotes to create a string, it first looks for string with the same value in the string pool. If found it just returns the reference else it creates a new string in the pool & then returns the reference.

→ By using new operator, we force String class to create a new string object in heap space. We can use intern() method to put it into the pool or refer to another string object from the string pool having the same value.

required: String pool is nothing but a storage area in Java heap where string literal stores. To decrease the no/ of string objects created in the JVM the String class keeps a pool of strings. When we create a string literal, the JVM first check that literal in the string pool.

(5) String s = new String("abc");
String s1 = "abc";
how many string are created?

⇒ two string are created.
① inside heap
② inside string pool

⑥ What is the use of String intern() Method. Example required.

→ Java String intern() method returns the interned String. It returns the canonical representation of String. It can be used to return string from memory, if it is created by new keyword. It creates exact copy of heap string object in String constant pool.

→ String interning is a method of String only one copy of each distinct string value, which must be immutable. By applying String.intern() on a couple of strings will ensure that all strings having the same contents share the same memory.

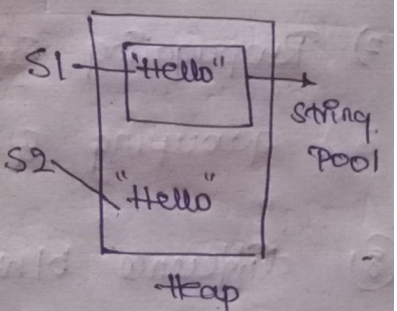
Ex: ① String S2 = new String ("Hello");

String S1 = "Hello";

S2.intern();

S.O.P (S1 == S2);

O/P: true



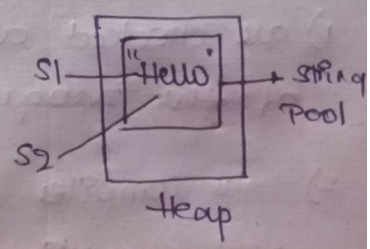
String ⇓ Interning

Ex② String S1 = "Hello";

String S2 = new String ("Hello").intern();

S.O.P (S1 == S2);

O/P: true



Exception Handling

(7) Explain the importance of exception handling. Why is it required?

defn: An Exception is an event, which occurs during the execution of a program, that disrupts the normal flow of program's instructions.

→ when an error occurs within a method creates a object and hands it off to the runtime s/m. The object, called an exception object, contains info-mation about the error, including its type and the state of the program, when the error occured

→ Creating an exception object and handing it to the the runtime s/m is called throwing an exception

Advantage: 1) Separating error-handling code from "regular" code.

2) Propagating Error up the call stack.

3) grouping & differentiating Error types.

(8) difference b/w checked & unchecked exceptions?

checked Exception	unchecked exceptions
1) are checked at runtime of the program.	1) are checked at compile time of the program.
2) the compiler checks a check-ed exception.	2) The compiler does not check these types of exception.

③ These types of exception can be handled at the time of compilation.

④ They are the sub-class of the exception class

⑤ Here, the JVM needs the exception to catch & handle

⑥ Ex: file Not Found Exception.
No Such Field Exception
Interrupted Exception
No Such Method Exception
class Not Found Exception.

③ These types of exceptions cannot be a catch & handle at the time of compilation bec/ they get generated by the mistakes in the program

④ They are runtime exception & hence are not a part of the Exception class.

⑤ Here, the JVM does not require the exception to catch and handle

⑥ Ex: NO Such Element Exception
Undeclared Throwable Exception
Empty stack Exception.
Arithmetic Exception.
Null Pointer Exception.
Array Index out of Bounds Exc/
Security Exception.

⑨ difference b/w Error & Exceptions.

Error	Exception
1). An error is something that most of the time you cannot handle it.	1) Exceptions are those which can be handled at the runtime.
2) Type: classified as an unchecked type	2) classified as checked and unchecked.
3) Package: It belongs to Java.lang.error	3) It belongs to Java.lang. Exception.
4) It is irrecoverable	4) It is recoverable

⑤ It can't be occur at compile time.	⑤ It can occur at run time compile time both.
⑥ Example: OutOfMemoryError, IOError	⑥ Example: NullPointerException SqlException.

⑩ Are errors throwable?

→ An error is a subclass of Throwable that indicates serious problems that a reasonable application should not try to catch.

→ Most such errors are abnormal conditions. the ThreadDeath error, though a "normal" condition, is also a subclass of Error bec/ most applications should not try to catch it.

⑪ difference b/w throw & throws

throw	throws
1) throw keyword is used inside a func/. it is used when it is required to throw an Exception logically	1) throws keyword is in the func/ signature. It is used when the func/ has some stmts that can lead to some exceptions.
2) throw is used to throw an exception explicitly. It can throw only one exception at a time	2) throws can be used to declare multiple exceptions, separated by comma. When -ever exception occurs, if matched with the declared ones, is thrown automatically then.
Ex: throw new IOException();	Ex: Void Demo() throws ArithmaticExc -eption, NullPointerException { }

③ Syntax wise throw keyword is followed by the instance variable.	③ Syntax wise throws keyword is followed by exception class names.
④ Checked exception cannot be propagated using throw only unchecked exception can be propagated using throw	④ For the propagation checked exception must use throws keyword followed by specific exception class name.

⑫ What is "finally" keyword used for?

→ The finally keyword is used in association with a try/catch block and guarantees that a section of code will be executed, even if an exception is thrown.

↳ the finally block will be executed after the try & catch blocks, but before control transfers back to its origin.

⑬ Give example for multi-catch syntax?

→ Ex: Public class MultipleCatchBlock1 {

```
Public static void main (String[] args) {
try {
    int a[] = new int[5];
     a[5] = 30/0; }
catch (ArithmeticException e)
{
System.out.println (" Arithmetic Exception occurs ");
    }
catch ( ArrayIndexOutOfBoundsException e )
{   S.O.P (" ArrayIndexOutOfBounds Exception occurs "); }
```

```
catch (Exception e)
{
    S.O.P (" Parent Exception occurs ");
}   S.O.P ("rest of the code");
}
}
```

O/P: Arithmetic Exception occurs.
rest of the code.