1. Class loader:

System Loader (classpath)->Extension Class loader->Bootstrap Class Loader (core java files)

2. primitive type: byte, short, int, long, float, double, char, boolean

3. wrapper class: wrapper around a data type and give it an object appearance.

why do we need wrapper class?

Null is a possible.

Use it in Collection because collection only accept object.

support object creation from another type - Integer number2 = new Integer("50"); //String

Ways to create wrapper class:

using a wrapper class constructor. Integer number2 = new Integer("50");

Using valueof. Integer number2 = Integer.valueof("2");

The different is constructor is creating object and valueof is using static method.

4. Auto boxing - automatic conversuon that Java compiler between the primitive and object wrapper class.

example 1:

Integer a= 9;

Integer b=9;

a=b

example 2:

Integer a= new Integer(10);

Integer b= new Integer(10);

a!=b;

a.equals(b) //true

5. Casting – convert primitive data type from one to another

implicit casting – done by compiler, store small to large

int a=100;

long b=a; // implicit casting

explicit casting - – done by code, store large to small

long number1= 1234;

int number2= (int) number1; //explicit casting

6. String – String is immutable

String is store at “String constant pool” which is inside the Heap memory, that is reused. For example String a= 11; However if the new operator is used to create a object, the new object is created on heap that is no reused value.

Avoid to use String concatenation in the loop because that new concatenation will create a new object and it may cause performance issue. To resolve this issue, we need to use StringBuffer

StringBuffer a = new StringBuffer(“value1”);

StringBuffer b = new StringBuffer(“value2”);

For (int i=1;, i<10000;i++)

a.append(b);

7. Different between String and StringBuffer

String is immutable and StringBuffer is use to present value that can be modified.

StringBuffer has between performance.

Both String and StringBuffer is thread safe.

StringBuffer is use synchronized key word on this method.

8 Different between StringBuild and StringBuffer

StringBuild is not thread secure.

8. String methods.

Str.length()

Str.charAt()

Str.concat(str2)

Str.printf

Str.format

Str.toString

Str.copyOfValue

Str.endWith

Str.indexOf

Str.matches

Str.replace

9. Collections

Array is not dynamic and collection is dynamic. And collection allow to add/delete elements and also allow to host another operation. The interface in collection hierarchy

Interface Collection<E> extends iterable <E> {

}

//unique thing.

Interface Set<E> extends Collection<E> {

}

//List things.

Interface List<E> extends Collection<E> {

}

//Arrange in order of processing.

Interface Queue<E> extends Collection<E> {

}

//no extends to collection, it maps to Key => Value {(“key 1”, value 1), {(“key 2”, value 2)}

Interface Maps<v,k> {

Create a collection

Collection collection = new ArrayList();

**Collection Subtypes**

The following interfaces (collection types) extends the *Java* *Collection* interface:

* [**List**](http://tutorials.jenkov.com/java-collections/list.html)
* [**Set**](http://tutorials.jenkov.com/java-collections/set.html)
* [**SortedSet**](http://tutorials.jenkov.com/java-collections/sortedset.html)
* [**NavigableSet**](http://tutorials.jenkov.com/java-collections/navigableset.html)
* [**Queue**](http://tutorials.jenkov.com/java-collections/queue.html)
* [**Deque**](http://tutorials.jenkov.com/java-collections/deque.html)

10. Interface - An interface is a completely "abstract class" that is used to group related methods with empty bodies.

// interface

interface Animal {

public void animalSound(); // interface method (does not have a body)

public void sleep(); // interface method (does not have a body)

}

To access the interface methods, the interface must be "implemented" (kinda like inherited) by another class with the implements keyword (instead of extends).

// Pig "implements" the Animal interface

class Pig implements Animal {

public void animalSound() {

// The body of animalSound() is provided here

System.out.println("The pig says: wee wee");

}

public void sleep() {

// The body of sleep() is provided here

System.out.println("Zzz");

}

}