Agenda

- Revision
    - Wrapper
    - Boxing & Unboxing
    - Reference
    - Packages
    - Yesterday's revision
- Final
- Static

Revision

- Java have provided classes to every primitive types
- these are called as wrapper classes.
- Why to use them?
    - to convert primitive type to reference type.
    - List list;
    - To use helper/utility methods

```
// Boxing
int num1 = 10;
Integer i1 = Integer.valueOf(num1); // boxing
Intger i2 = num1; /// auto-boxing

// UnBoxing
String []args;
// converting from reference types into primitive types
int num1 = Integer.parseInt(args[0]); // un-boxing
```

Reference

- class variable that we create is called as reference.
- reference points at an instance of a class
- reference holds the object

Packages

- It is a container to organise the code, also use dto avoid name ambugity for the classes.
- sunbeam.com -> eAttendance
- package name -> com.sunbeam.eattendance

Final Keyword (Demo01)

- const is reseverd keyword in java, however it is rarely used.
- Instead of const java have provided a keyword called as final.
- We can make
    - 1. variables as final
    - 2. fields as final
    - 3. method as final;
    - 4. class as final;

Final Variables;

- If we declare variables as final then once we assign value to them, the value cannot be changed.
- we can make the local variables as final.
- final varaibles can be only declared and can be assigned with a value later.
- however once assigned then you cannot change it.

```
    //final int num1 = 10;

    // OR
        final int num1;
        num1 = 10;

        //num1 = 20;// NOT OK

    final Test test = new Test();
    test.display();

    test = new Test(); // NOT OK
```

Final Fields

- We can also declare the fields as final
- when we declred field as final we can inititialize them in the follwing ways
    - 1. in field initializer
    - 2. in object initializer
    - 3. inside constructor
- once the fields are initialized we cannot change the value.

```
Test {
    final private int field1 = 1001; // field initializer
    final private int field2;
    final private int field3;

    // Object initializer
    {
        field2 = 2001;
    }

    public Test() {
```

```
        field3 = 3001;
    }
}
```

```
class BankAccount{
  final int accno;
  String name;
  double balance;

  // constructor
  BankAccount(int accno){
    this.accno = accno;
  }
}

class Circle{
int radius;
final double PI = 3.14; // field initializer
}
```

Static Keyword

- static means shared varaible
- in java we cannot cretae static local varaibles
- In java we can use static for
    - 1. fields
    - 2. methods
    - 3. block
    - 4. import

Static Field

- static fields get the memory on the Method Area
- static fields are class level members and not the object members.
- i.e they are designed to be shared across multiple objects.
- static fields can be access using className and . operator outside the class if they are made as public.
- however we can also access them using object using . operator. As it is misleading it is better to avoid it.
- memory for the static fields are given at the time of class loading in the JVM.
- memory for the static fields are given only once. All the objects will share the same static field.

```
class Circle{
int radius;
static final double PI = 3.14; // field initializer
}
```

Static Block

- we can initialize the static fields in 2 ways
    - 1. field initializer
    - 2. static block
- static block is same as that of object inititializer block,but with static keyword.
- static block executes only once.

static methods

- if we want to call the methods of the class on classname without creating the object then make such methods as static.
- static methods can access only static fields. it cannot access non static fields inside them.
- static methods do not get this reference and thats why we cannot access non static fields inside them.
- static methods can be access using className and . operator outside the class if they are made as public.
- however we can also access them using object using . operator. As it is misleading it is better to avoid it.

Examples of static Field and methods

Fields

- Integer.SIZE (Integer is a class SIZE is a static field)
- Integer.MAX_VALUE (Integer is a class MAX_VALUE is a static field)
- System.out (System is a class out is a static field)
- System.in (System is a class in is a static field)

Methods

- main() (In java by default main method is static)
- Integer.parseInt() (Integer is a class parseInt() is a static method)
- Integer.valueOf() (Integer is a class valueOf() is a static method)

static import (Demo02->Program02)

- In java, we can access the static methods of same class directly inside aother static methods.
- However to access the static methods of other class from different package we have to use classname and . operator.
- If we also want to access the static members of class from different packages direcetly then we should use static import.

Lab Work

- Singleton
- OOP
    - Association
        - Aggegration

        - Composition
    - Inheritance
        - types of inheritance