This Java notes prepared by:
Sadar Singh Bhukya
Data Manipulation Logics(DML)
Hyderabad-Telangana-India
91-9701704230

Java:

- It is an object oriented programming language
- It is high reusable language

1. Class structure:→It is a template ,contains variables and methods

```
class Classname{
}
```

example-->

```
class Sadar{

}
```

2. Method structure: action name:→ specific code for specific purpose with name

```
void methodname(){
}
```

example-->

```
void add{

}
```

3. Variable:→ Declaration memory name to store&retrieve data.

Int salary;
salary→ variable name: field
nameint --> Data type

4. Entering value in variable

salary=50000;

5. Object creation: **constructor**→ creating object for particular class to use variables and methods of class

Classname objname=new Classsname();

example-->

```
Sadar objsadar=new Sadar();
```

Selecting variable and method with object

objname.varaiable
objname.method

example-->

```
objsadar.add();
```

6. How do u identify methods and variables while using?

Object.add()→method with bracket
Object.add→variable without bracket

Example of class with variable and method:

```
class Hdfc {
        int salary=50000;
        void add(){

        }
    }
```

6. **Access modifiers**: it describes accessibility of the data

a.**private**: accessibility data within class
b.**default**: accessibility data within package
c.**public**: accessibility data within project
d.**protected**: accessibility data source and destination classs in inheritance concept

ex:
```
public class University{
public int salry;
public void add(){
}
}
```

7. Methods types:

1. NonReturntype: void
a. without parameter
Student object=new Student();
object.add();
b. with parameter
object.add(2,3);

2. Return type: int,String,char….etc

a. without parameter
int total=object.add()
system.out.println(total);

b. with parameter
int total=object.add(2,3)
system.out.println(total);

NonReturntype: void

```
class Student{
void add() {
int a=10;
int b=5;
int t=a+b;
system.out.println(t);
        }

void add(int a,int b) {
int t=a+b;
system.out.println(t);
        }

}
```

Returntype: int

```
class Student{
int add() {
int a=10;
int b=5;
int t=a+b;
return t;
        }

Int  add(int a,int b) {
int t=a+b;
return t;
        }

}
```

8.OOPS: Reusability of the data, sharing the data
1.Inheritance: Acquiring one class attribute to another class

Create class1 ..... create classe2 ..... Main execution

```
class SSB {
void  add(){
}
}
```

```
class RRB extends SSB {
void sub(){

}
}
```

```
Result: Create an object for RRB
RRB obj=new RRB();
 obj.add();
obj.sub();
```

// note: two methods are available
with child class object

When do use super and this key words?

Super → represents super class variable and method

this → represents same class variable and method

execution code

```
RRR obj=new RRR();
    Obj.add();
```

```
class SSR {
int a=10;
void  ok(){
system.out.print("ok");

}
}
```

```
class RRR extends SSR  {
int a=20;
void add(){
int t= super.a+this.a;
system.out.print(t);
super.ok();
}
```

2.Interface: Defined methods and variables of interface to implements in different classes

Create interface ----------create class implements interface ---Main Execution

```
public interface RBI {
public void loan();
}
```

```
class SSB implements RBI{
void loan(){
    }
}
```

Result: Create an object combination of interface and class and observe while selecting methods

```
RBI obj=new SSB;
obj.loan();
//  note: method showing of  interface
```

3.Abstract class

It is also like as interface but class with abstract key word is used and abstract class extends in classes

Create class with abstract option-----create class extends RBI--------- -----execution

```
public abstract class RBI {
public abstract void add();
}
```

```
class  SSB extends RBI {
    void add(){
                }
}
```

Result: Create an object combination of abstract class and class and observe while selecting methods

```
RBI obj=new SSB()
obj.add();
//  note: method showing in  abstract class
```

4.Polymorphism: Reusability of methods

a. **Overloading:** same method name with different parameters and different data types

Create class with same methods name ---------- execution

```
class  calculator  {
void add(){
int   a=10;
int   b=20;
int t=a+b;
system.out.println(t);
}
void add(int a,int b){
int t=a+b;
system.out.println(t);
}

void add(float a,float b){
float t=a+b;
system.out.println(t);
}
}
```

Result: Create an object ,select methods and select methods types

```
calculator obj=new calculator();
obj.add();
obj.add(2,3);
obj.add(5f,9f);
```

b. **Overriding:** creating object with inherited classes

Create class with one method--- create another class same method name and extends it—execution

```
class  Parent  {
void add(){
int   a=10;
int   b=20;
int t=a+b;
system.out.println(t)  ;
}
)
```

```
class  Child  extends  Parent{
void add(){
int a=400;
int b=500;
int t=a+b;
system.out.println(t)  ;
}
)
```

Result: Create an object with parent and child class combination and observe while selecting the method

```
Parent obj=new Child ();
obj.add();
//note: showing parent class but it will execute child class method
```

5.Encapsulation: data hiding within one class, it can be achieved by using public methods to assigning and retrieving private variables of class

Create class with private variable and public method----------Execution

```
class  University {
private int a;
private int b;
public void set(int i , int j){
 a=i ;
b=j;
}
public void add(){
int t = a+b ;
System.out.println(t) ;
}
```

```
public int getvaluea(){
 int t=a;
return t;
}

public int getvalueb(){
 int p=b;
return p;
}
}
```

Result: Create an object with class

```
University obj=new University ();
// Assign private variable by using public methods
obj.set(100,200);
obj.add();
//Getting private  variable values  by using public methods
```

```
int i=obj. getvaluea();
 int j=obj. getvalueb();
System.out.println(i) ;
System.out.println(j) ;
System.out.println(i+j) ;
```

9.Construction with parameter: → passing value from constructor to class variable.
Create class and method also same class name with parameter variables and public method

```
class Calc {
int a;
int b;
void add(){
int t=a+b;
system.out.println(t);
}
public Calc(int d,int e){
a=d;
b=e;
}
)
```

Execution

```
Calc obj=new Calc(10,20);
Obj.add();
```

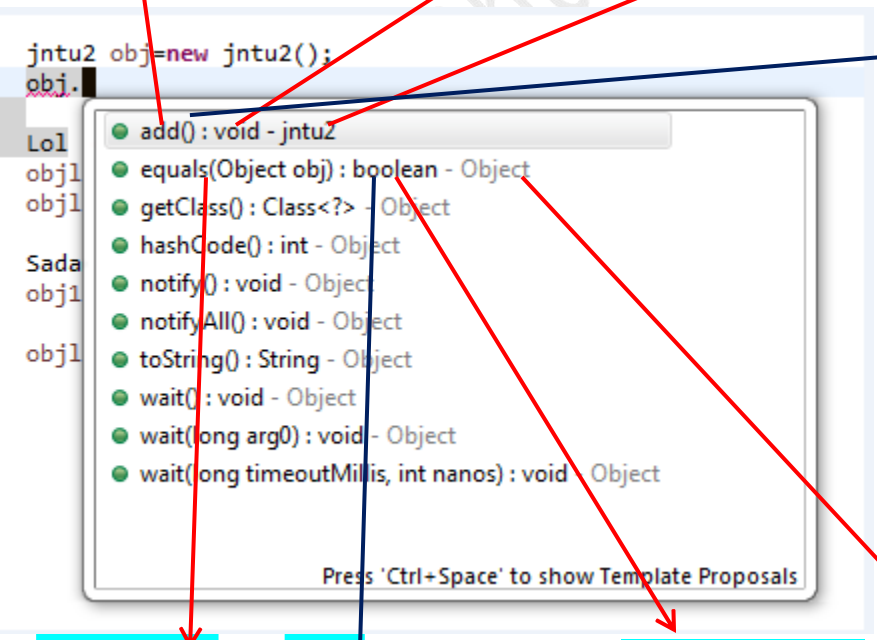
10.How many ways do you create an object?

1. **class and class**
2. **Interface and class**
3. **parent class and child class(overriding)**
4. **abstract class and class**

11.Methods description while selecting:

1.add method without parameter and **non-return type** in **Jntu2 class**

How to use NonReturn type
Obj.add();

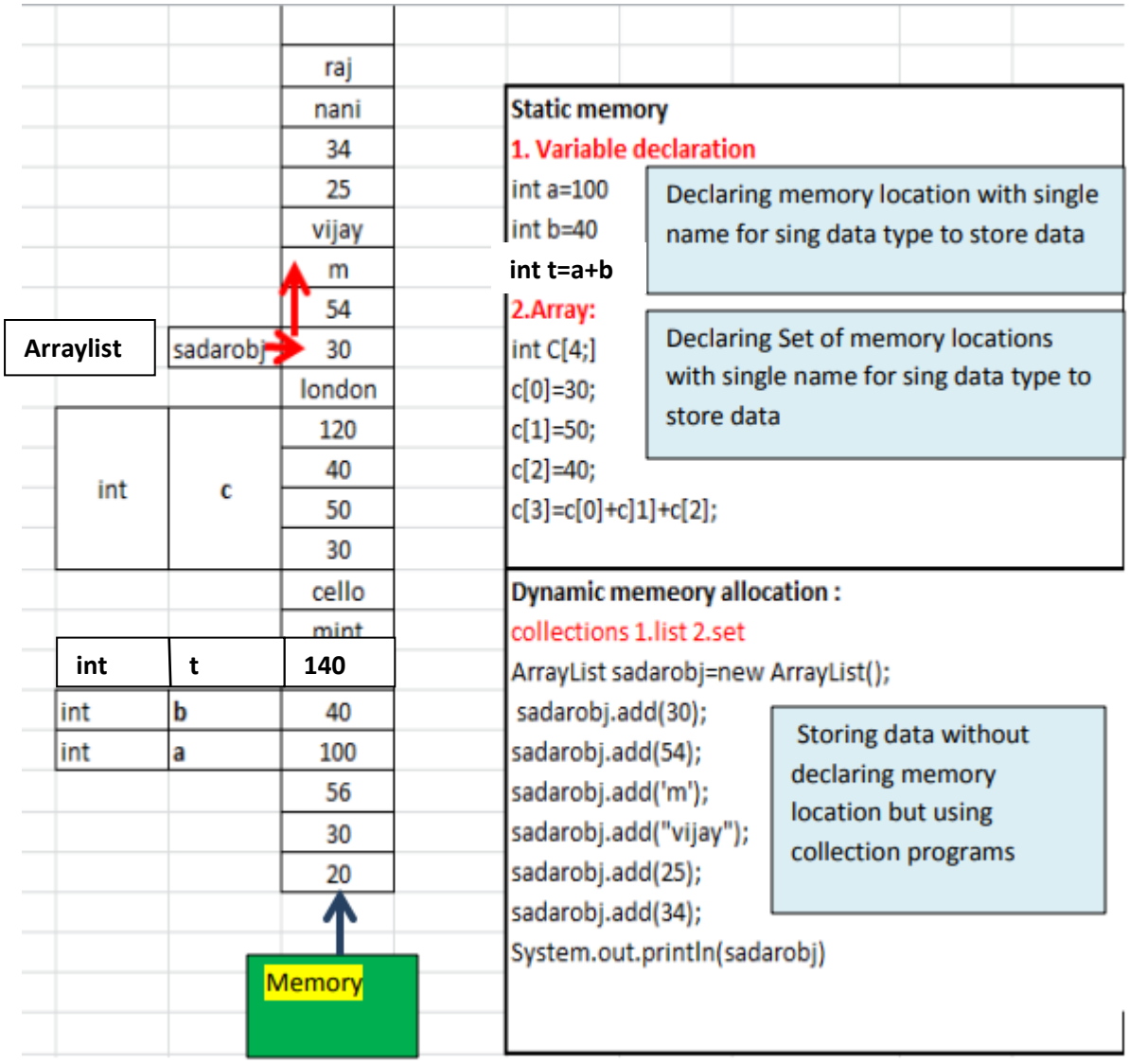


2. equals method with **Object** type parameter and **boolean return type** in **Object class**

How to use return type method

```
boolean t= obj.equals("value");
System.out.println(t);
```

Memory allocation and variable declaration types



13.Static memory allocation programs:

→Variable declaration and Array both of them are called Static memory allocation programs

Sample code with variable:

```
int a= 20;
int  b=30;
int  c=40
int t=a+b+c;
```

Sample code with Array:

```
int[] a=new int[4];
a[0]=20;
a[1]=30;
a[2]=40;
a[3]== a[0]+ a[1]+a[2];
```

14.Dynamic memory allocation programs (Collections):
→not declaring memory as a variable but storing the values by using the collection class object

Example code:

```
ArrayList obj=new ArrayList();
obj.add("sadar");
obj.add(234);
obj.add('m');
system.out.println(obj);
```

15.Dynamic memory allocation types (Collections):

1. **List 2. Set**

16.Exception Handling:
→ Guessing expected errors for failure of code execution

17.Exception Handling keywords:
1.try
2.catch
3.finally
4.throw—using with method execution
5.throws—using with method signature

Wish you all the best!
This Java notes prepared by:

Sadar Singh Bhukya

Data Manipulation Logics(DML)
Hyderabad-Telangana-India
91-9701704230