**Collection Framework – Part\_14**

* **Hashtable:**

1. The underlying data structure for Hashtable is Hashtable.
2. Insertion order is not preserved and it is based on hashCode of keys.
3. Duplicate keys are not allowed, and values can be duplicated.
4. Heterogeneous objects are allowed for both keys and values.
5. Null is not allowed for both key and value, other we will get runtime exception saying: NullPointerException
6. It implements Serializable and Cloneable interfaces but not RandomAccess.
7. Every method present in Hashtable is synchronized and hence Hashtable object is thread-safe.
8. Hashtable is the best choice if our frequent operation is search operation.

* **Constructors:**

Hashtable h = new Hashtable();

Creates an empty Hashtable object with default initial capacity 11 and default fill ration 0.75.

Hashtable h = new Hashtable(int initialCapacity);

Hashtable h = new Hashtable(int initialCapacity, float fillRatio);

Hashtable h = new Hashtable(Map m);

* **Example:**

import java.util.\*;

class HashtableDemo{

public static void main(String[] args){

Hashtable h = new Hashtable();

h.put(new Temp(5), “A”);

h.put(new Temp(2), “B”);

h.put(new Temp(6), “C”);

h.put(new Temp(15), “D”);

h.put(new Temp(23), “E”);

h.put(new Temp(16), null);

h.put(“durga”, null); // NPE

System.out.println(h);

}

}

class Temp{

int i;

Temp(int i){

This.i = i;

}

public int hashCode(){

return i;

}

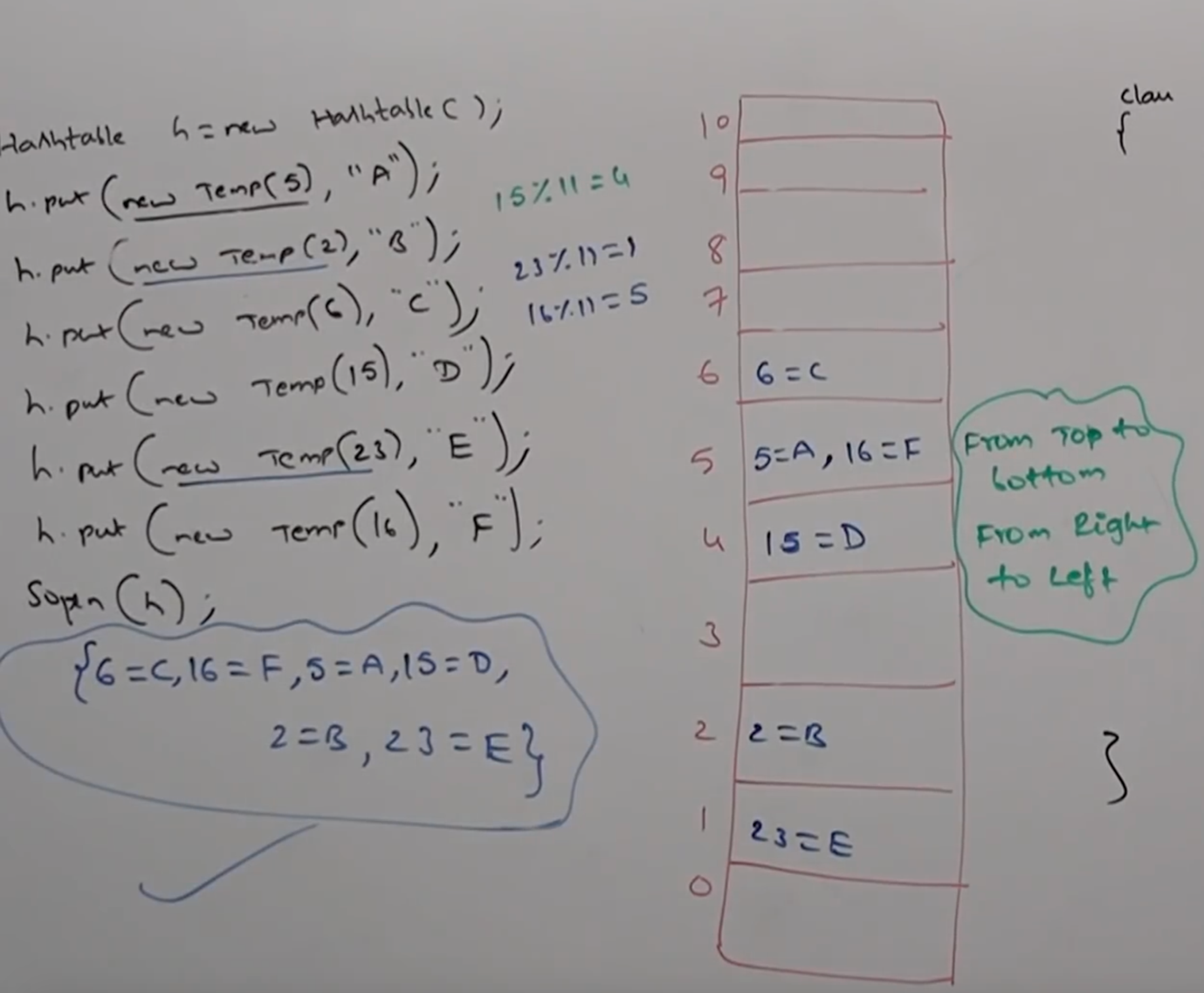
public String toString(){

return i+””;

}

}

* **Diagrammatic Representation:**

****

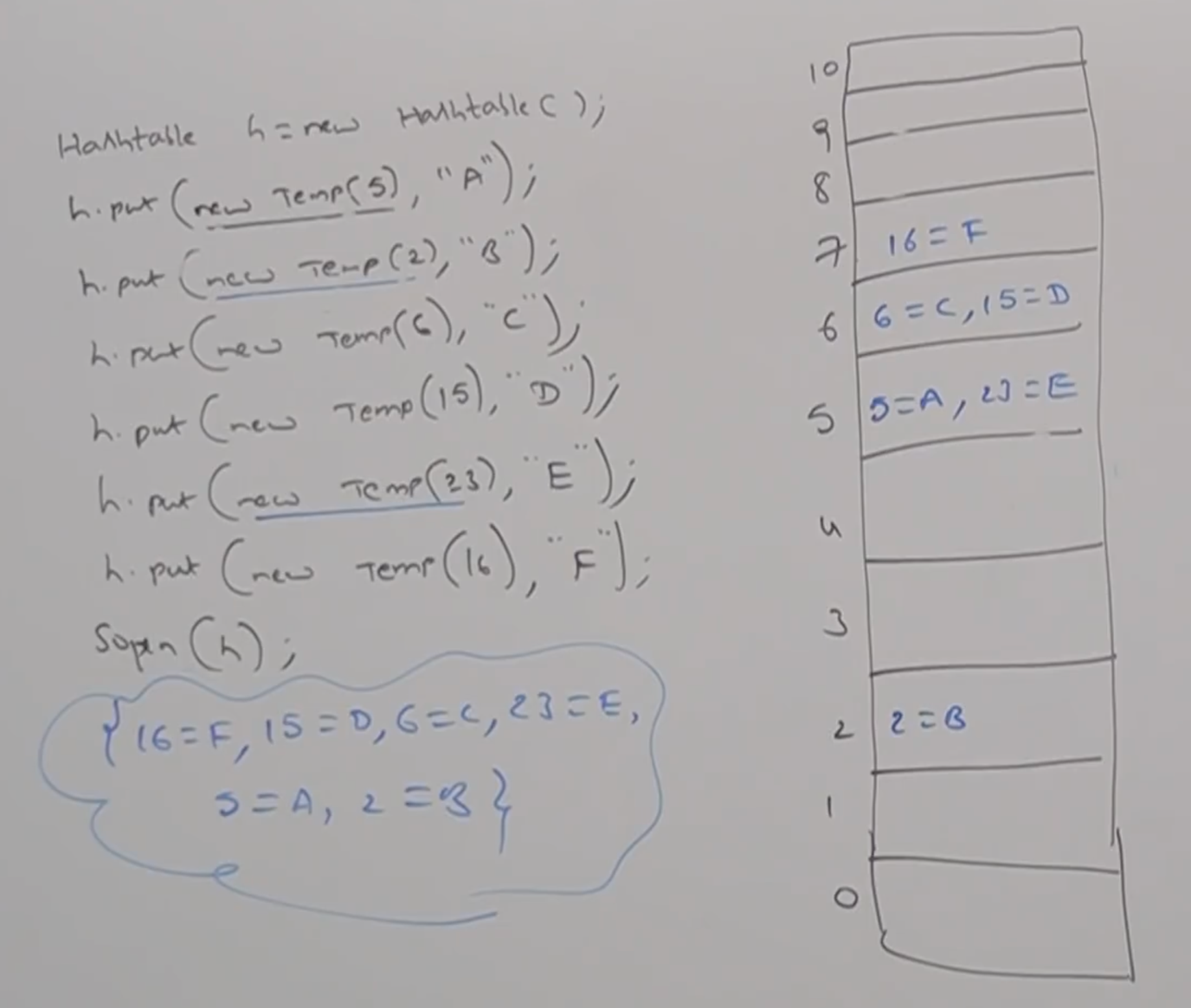
If we change hashCode() method of Temp class as

public int hashCode(){

return i%9;

}

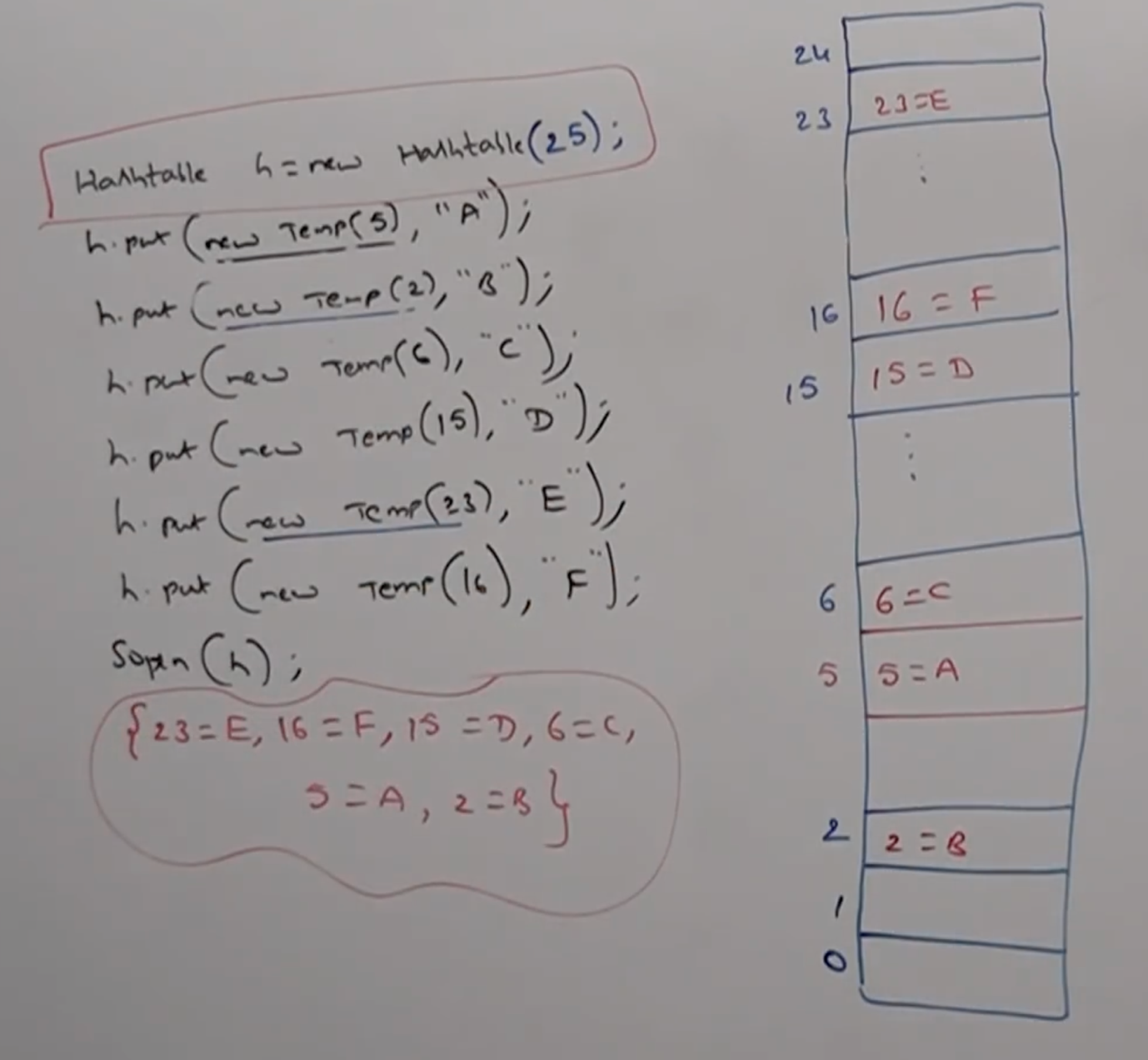
Now the elements placement and output changes as shown below.



If we initial capacity as 25, that is

Hashtable h = new Hashtable(25);

And the hashCode() method as first variant, that is just return i. Then we will get the following output.



* **Properties:**

In our program if anything which changes frequently (like username, password, mailids, mobilenumber etc) are not recommended to hard-code in Java program. Because, if there is any change to reflect that change recompition, rebuild, redeploy application are required even sometimes server restart also required. Which creates a big business impact to the client.

We can overcome this problem by using properties file, such type of variable things we have to configure in the properties file. From that properties file we have to read into Java program and we can use those properties.

The main advantage of this approach is, if there is a change in properties file to reflect that change just redeployment is enough, which won’t create any business impact to the client.

We can use Java properties object to hold properties which are coming from properties file.

In normal Map (like HasMap, Hashtable, TreeMap) key and value can be any type, but in the case of Properties key and value should be String type.

* **Constructor:**

Properties p = new Properties();

* **Methods:**

String setProperty(String pname, String pvalue)

To set a new Property. If the specified property already available then old value will be replaced with new value and returns old value.

String getProperty(String pname);

To get value associated with the specified property. If the specified property not available, then this method returns null.

Enumeration propertyNames();

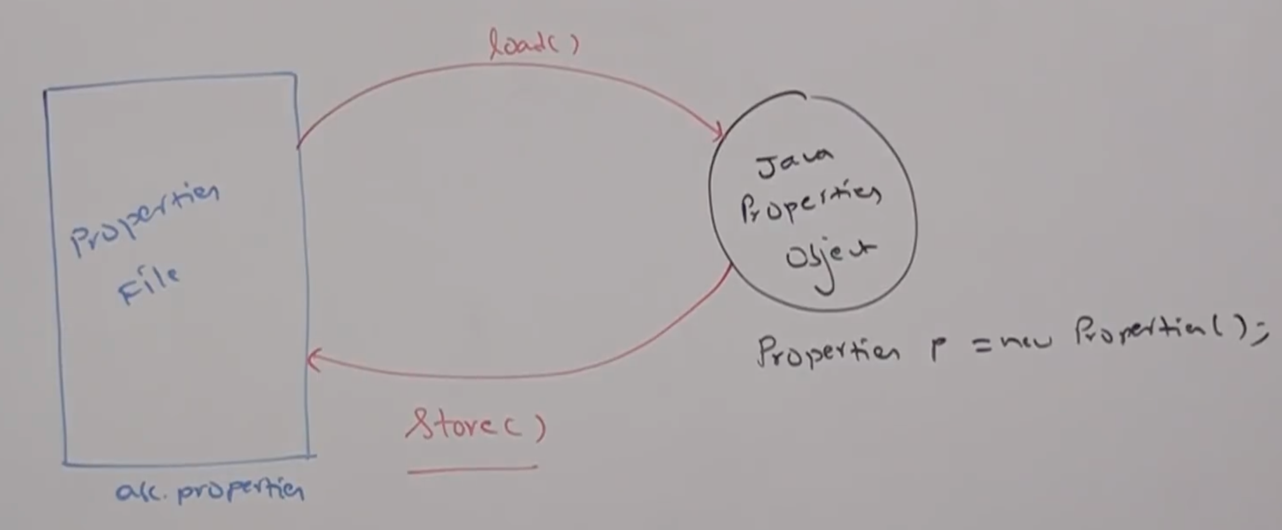
Returns all property names present in properties object.

void load(InputStream is)

To load properties from properties file into Java properties object.

void store(OutputStream os, String comment);

To store properties from Java properties object into properties file.



* **Example:**

import java.util.\*;

import java.io.\*;

class PropertiesDemo{

public static void main(String[] args) throws Exception{

Properties p = new Properties();

FileInputStream fis = new FileInputStream(“abc.properties”);

p.load(fis);

System.out.println(p); // {PN=PV}

String s = p.getProperty(“venki”);

System.out.println(p);

p.setProperty(“nag”);

FileOutputStream fos = new FileOutputStream(“abc.properties”);

p.store(fos, “Updated by Durga for SCJP demo class”);

}

}

Properties file:

abc.properties:

user=scott

pass=tiger

venki=999

* **Example\_02:**

class PropertiesDemo2{

public static void main(String[] args){

Properties p = new Properties();

FileInputStream fis = new FileInputStream(“db.properties”);

p.load(fis);

String url = p.getProperty(“url”);

String user = p.getProperty(“user”);

String pwd = p.getProperty(“pwd”);

Connection con = DM.getConnection(url, user, pwd);

}

}