

ЕДНОБАЗОВА ЙЕРАРХИЯ

ЛЕКЦИОНЕН КУРС “ПРОГРАМИРАНЕ НА JAVA”



СТРУКТУРА НА ЛЕКЦИЯТА

- Преглед на класа Object
- Примери

ЕДНОБАЗОВА ЙЕРАРХИЯ

- Един от значимите спорни въпроси в ООП е дали всички класове трябва задължително да наследяват един базов клас
 - В Java отговорът е „да“
 - Базовият клас Object
- За всички обекти в еднобазова йерархия може да се гарантира
 - Наличие на определена обща функционалност
 - Изпълнение на основни операции
 - Примери:
 - Garbage collector
 - Обработка на изключения
- Еднобазовата йерархия (заедно с хийпа) опростява предаването на параметри

МНОГОКРАТНО ИЗПОЛЗВАЕМИ КОНТЕЙНЕРИ

- Еднотазовата йерархия означава, че всичко е от клас Object
- Така, един контейнер, съдържащ обекти от този клас, може да съхранява всеки тип обекти
 - Това прави контейнерите многократно използвани
- За да се използва един такъв контейнер, просто добавяме обектни референции към него и по-късно ги изискваме отново
 - При добавяне на обектни референции се извършва преобразуване нагоре до Object
 - Така губим идентичността на обектите
 - Когато ги извличаме обратно, получаваме референции към Object, а не към типа
 - За да превърнем обратно използваме отново преобразуване – този път надолу по йерархията

ВЪВЕДЕНИЕ

- В Java, класът Object е коренът на всяка йерархия
 - Т.е., всеки клас наследява директно или индиректно Object
- **Object**: предоставя неголям брой методи, целесъобразни за използване от всички обекти
 - Т.е., много общи
- Намира се в пакета java.lang
- Притежава един прост конструктор
 - Object()

ПРЕГЛЕД НА МЕТОДИТЕ

Тип	Метод
protected Object	clone() : създава и връща копие на обекта
boolean	equals(Object obj) : индикира дали един обект е еквивалентен на друг
protected void	finalize() : извиква се от garbage collector, когато установи, че няма референции към обекта
Class<?>	getClass() : връща runtime класа на обекта
int	hashCode() : връща хеш код на обекта
void	notify() : събужда отделна нишка, която чака за изпълнение
void	notifyAll() : събужда всички нишки, които чакат за изпълнение
String	toString() : връща string, представящ обекта
void	wait() : причинява актуалната нишка да чака, докато друга нишка извика notify() метода или notifyAll() метода за този обект
void	wait(long timeout) : причинява актуалната нишка да чака, докато друга нишка извика notify() метода или notifyAll() метода за този обект или изтича зададеното време
void	wait(long timeout, int nanos) : причинява актуалната нишка да чака, докато друга нишка извика notify() метода или notifyAll() метода за този обект или изтича зададеното време или друга нишка прекъсва актуалната

ПРИМЕР

```
import java.util.ArrayList;

public class ObjectDemo {

    public static void main(String[] args) {

        // get an integer
        Integer i = new Integer(50);

        // get a list
        ArrayList list = new ArrayList();

        // add some elements in list
        list.add(50);
        list.add("Hello World");

        // print their string representation
        System.out.println(" " + i.toString());
        System.out.println(" " + list.toString());
    }
}
```

- Демонстрира toString()
- Извежда състоянията на обектите
- Обикновено, методът се препокрива в наследените класове

ПРИМЕР

?

Результат

```
import java.util.ArrayList;

public class ObjectDemo {

    public static void main(String[] args) {

        // get an integer
        Integer i = new Integer(50);

        // get a list
        ArrayList list = new ArrayList();

        // add some elements in list
        list.add(50);
        list.add("Hello World");

        // print their string representation
        System.out.println(" " + i.toString());
        System.out.println(" " + list.toString());
    }
}
```

50

[50, Hello World]

ПРИМЕР

```
import java.util.GregorianCalendar;

public class ObjectDemo1 {

    public static void main(String[] args) {

        // create a gregorian calendar, which is an object
        GregorianCalendar cal = new GregorianCalendar();

        // clone object cal into object y
        GregorianCalendar y = (GregorianCalendar) cal.clone();

        // print both cal and y
        System.out.println(" " + cal.getTime());
        System.out.println(" " + y.getTime());

    }
}
```

Демонстрира метода clone()

ПРИМЕР

?

Результат

```
import java.util.GregorianCalendar;

public class ObjectDemo1 {

    public static void main(String[] args) {

        // create a gregorian calendar, which is an object
        GregorianCalendar cal = new GregorianCalendar();

        // clone object cal into object y
        GregorianCalendar y = (GregorianCalendar) cal.clone();

        // print both cal and y
        System.out.println(" " + cal.getTime());
        System.out.println(" " + y.getTime());

    }
}
```

Sat Feb 07 08:09:07 EET 2015
Sat Feb 07 08:09:07 EET 2015

ПРИМЕР

```
public class ObjectDemo2 {  
  
    public static void main(String[] args) {  
  
        // get an integer, which is an object  
        Integer x = new Integer(50);  
  
        // get a float, which is an object as well  
        Float y = new Float(50f);  
  
        // check if these are equal, which is  
        // false since they are different class  
        System.out.println("'" + x.equals(y));  
  
        // check if x is equal with another int 50  
        System.out.println("'" + x.equals(50));  
  
    }  
}
```

Демонстрира метода equals()

ПРИМЕР

?

Результат

```
public class ObjectDemo2 {  
  
    public static void main(String[] args) {  
  
        // get an integer, which is an object  
        Integer x = new Integer(50);  
  
        // get a float, which is an object as well  
        Float y = new Float(50f);  
  
        // check if these are equal, which is  
        // false since they are different class  
        System.out.println("" + x.equals(y));  
  
        // check if x is equal with another int 50  
        System.out.println("" + x.equals(50));  
  
    }  
}
```

false
true

ПРИМЕР

```
import java.util.GregorianCalendar;

public class ObjectDemo3 {

    public static void main(String[] args) {

        // create a new ObjectDemo object
        GregorianCalendar cal = new GregorianCalendar();

        // print current time
        System.out.println("" + cal.getTime());

        // print a hashCode for cal
        System.out.println("" + cal.hashCode());

        // create a new Integer
        Integer i = new Integer(5);

        // print i
        System.out.println("" + i);

        // print hashCode for i
        System.out.println("" + i.hashCode());
    }
}
```

Демонстрира метода hashCode()



ПРИМЕР

```
import java.util.GregorianCalendar;

public class ObjectDemo {

    public static void main(String[] args) {

        // create a new ObjectDemo object
        GregorianCalendar cal = new
        GregorianCalendar();

        // print current time
        System.out.println("" + cal.getTime());

        // print a hashCode for cal
        System.out.println("" + cal.hashCode());

        // create a new Integer
        Integer i = new Integer(5);

        // print i
        System.out.println("" + i);

        // print hashCode for i
        System.out.println("" + i.hashCode());
    }
}
```



Резултат

```
Sat Feb 07 08:19:21 EET 2015
-830063227
5
5
```


ПРИМЕР

```
import java.util.*;
public class ObjectDemo extends
GregorianCalendar {

    public static void main(String[] args) {
        try {
            // create a new ObjectDemo object
            ObjectDemo cal = new ObjectDemo();

            // print current time
            System.out.println("'' + cal.getTime());

            // finalize cal
            System.out.println("Finalizing...");
            cal.finalize();
            System.out.println("Finalized.");

        } catch (Throwable ex) {
            ex.printStackTrace();
        }
    }
}
```

- Демонстрира метода finalize()
- Извиква се от garbage collector за един обект, когато garbage collection определи, че няма повече референции към този обект
- Подкласовете могат да препокриват този метод за извършване на завършващи операции

ПРИМЕР

?

Результат

1

```
import java.util.*;
public class ObjectDemo4 extends
GregorianCalendar {

    public static void main(String[] args) {
        try {
            // create a new ObjectDemo object
            ObjectDemo4 cal = new ObjectDemo4();

            // print current time
            System.out.println("" + cal.getTime());

            // finalize cal
            System.out.println("Finalizing...");
            cal.finalize();
            System.out.println("Finalized.");

        } catch (Throwable ex) {
            ex.printStackTrace();
        }
    }
}
```

```
Wed Feb 11 14:30:22 EET 2015
Finalizing...
Finalized.
```

БЛАГОДАРЯ ЗА ВНИМАНИЕТО!

КРАЙ “ЕДНОБАЗОВА ЙЕРАРХИЯ”

