Exceptions
November 30, 2007

Quick Review of Polymorphism & Polymorphic Variables

Class Hierarchy

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

type = A

Class Hierarchy

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

type = A & B

type = A & B & D

Class Hierarchy

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

type = A

type = A & B
Example: Letters class hierarchy

- A.java
- B.java
- C.java
- D.java
- E.java
- F.java
- G.java
- Driver.java

```java
public abstract class A {
    abstract void name();
}

public class B extends A {
    public void name() {
        System.out.println("B");
    }
    public void Bstuff() {
        System.out.println("BB");
    }
}

public class C extends A {
    public void name() {
        System.out.println("C");
    }
    public void Cname() {
        System.out.println("CC");
    }
}

public class D extends B {
    public void name() {
        System.out.println("D");
    }
    public void Bstuff() {
        System.out.println("DD");
    }
    public void Dstuff() {
        System.out.println("DDD");
    }
}

public class E extends B {
    public void name() {
        System.out.println("E");
    }
    public void Bstuff() {
        System.out.println("EE");
    }
    public void Estuff() {
        System.out.println("EEE");
    }
}
```
Quick Review of Last Lecture

Example: Animals class hierarchy

- Animal.java
- Cow.java
- Duck.java
- Dog.java
- Farm.java
You can use jGrasp to draw diagram like this one.

```java
public abstract class Animal
{
    abstract void makeSound();
}

public class Cow extends Animal
{
    public void makeSound()
    {
        System.out.println("Moo-Moo");
    }
}

public class Dog extends Animal
{
    public void makeSound()
    {
        System.out.println("Wuf-Wuf");
    }
}

public class Duck extends Animal
{
    public void makeSound()
    {
        System.out.println("Quack-Quack");
    }
}

public class Farm
{
    public static void main(String[] args)
    {
        Cow c = new Cow();
        Dog d = new Dog();
        Duck k = new Duck();
        c.makeSound();
        d.makeSound();
        k.makeSound();
    }
}

Result:
Moo-Moo
Wuf-Wuf
Quack-Quack
```

```java
public class Farm2
{
    public static void main(String[] args)
    {
        Animal[] a = new Animal[3];
        a[0] = new Cow();
        a[1] = new Dog();
        a[2] = new Duck();
        for(int i=0; i<a.length; i++)
            a[i].makeSound();
    }
}

Result:
Moo-Moo
Wuf-Wuf
Quack-Quack
```
We can do this...

```java
public abstract class Animal
{
    public abstract void makeSound();
    public void move()
    {
        System.out.println("walk");
    }
}

public abstract class Cow extends Animal
{
    public void makeSound()
    {
        System.out.println("Moo-Moo");
    }
}

public abstract class Dog extends Animal
{
    public void makeSound()
    {
        System.out.println("Wuf-Wuf");
    }
}

public abstract class Duck extends Animal
{
    public void makeSound()
    {
        System.out.println("Quack-Quack");
    }
}

public class Farm2b
{
    public static void main(String[] args)
    {
        Animal[] a = new Animal[3];
        a[0] = new Cow();
        a[1] = new Dog();
        a[2] = new Duck();

        for(int i=0; i<a.length; i++)
            a[i].move();
    }
}
```

Result:
walk
walk
walk

But if we add more classes to the Class hierarchy

```java
public abstract class Animal
{
    public abstract void makeSound();
    public void move()
    {
        System.out.println("walk");
    }
}

public abstract class Cow extends Animal
{
    public void makeSound()
    {
        System.out.println("Moo-Moo");
    }
}

public abstract class Dog extends Animal
{
    public void makeSound()
    {
        System.out.println("Wuf-Wuf");
    }
}

public abstract class Duck extends Animal
{
    public void makeSound()
    {
        System.out.println("Quack-Quack");
    }
}

public class Farm2b
{
    public static void main(String[] args)
    {
        Animal[] a = new Animal[3];
        a[0] = new Cow();
        a[1] = new Dog();
        a[2] = new Duck();

        for(int i=0; i<a.length; i++)
            a[i].move();
    }
}
```

Define a new method called move. It is not abstract and will be inherited by all children of Animal.

Override the move method defined in the Animal class.
public class Cow extends Animal
{
    public static void main(String[] args)
    {
        Animal[] a = new Animal[3];
        a[0] = new Cow();
        a[1] = new Dog();
        a[2] = new Duck();
        for(int i=0; i< a.length; i++)
            a[i].move();
    }
}

Result:
Walk
Walk
Fly

public abstract class Animal
{
    abstract void makeSound();
    public void move();
    { System.out.println("walk"); }
}

public class Cow extends Animal
{
    public void makeSound()
    {
        System.out.println("Moo-Moo");
    }
}

public class Dog extends Animal
{
    public void makeSound()
    {
        System.out.println("Quack-Quack");
    }
}

public class Duck extends Animal
{
    public void makeSound()
    {
        System.out.println("Quack-Quack");
    }
    public void dive()
    {
        System.out.println("Diving...");
    }
}

public class Farm2d
{
    public static void main(String[] args)
    {
        Animal[] a = new Animal[3];
        a[0] = new Cow();
        a[1] = new Dog();
        a[2] = new Duck();
        for(int i=0; i< a.length; i++)
            a[i].dive();
    }
}

Compile Error, since dive() is defined only for Duck objects and not for all objects derived from Animal.

This works OK, but requires a cast from a reference to Animal to a reference to Duck.

public class Farm2d
{
    public static void main(String[] args)
    {
        Animal[] a = new Animal[3];
        a[0] = new Cow();
        a[1] = new Dog();
        a[2] = new Duck();
        ((Duck)a[2]).dive();
    }
}

Result:
Diving...
Chapter 9
Section 9.3

public class Cow implements Animal
{
    public void makeSound()
    {
        System.out.println("Moo-Moo");
    }
}

public class Dog implements Animal
{
    public void makeSound()
    {
        System.out.println("Wuf-Wuf");
    }
}

public class Duck implements Animal
{
    public void makeSound()
    {
        System.out.println("Quack-Quack");
    }
}

public interface Animal
{
    public void makeSound();
    public void move();
}

public class Farm
{
    public static void main(String[] args)
    {
        Animal domestic;
        domestic = new Cow();
        domestic.makeSound();
        domestic = new Dog();
        domestic.makeSound();
        domestic = new Duck();
        domestic.makeSound();
    }
}

public class Farm2
{
    public static void main(String[] args)
    {
        Animal domestic;
        domestic = new Cow();
        domestic.makeSound();
        domestic = new Dog();
        domestic.move();
        domestic = new Duck();
        domestic.move();
    }
}

Result:
Moo-Moo
Wuf-Wuf
Quack-Quack

Define a new method an interface this method cannot be defined as in the previous example in an abstract class.

In this case Animal is an interface.

The Animals example with interfaces
implements
implements
implements

The only move() method that is different from fly.
Chapter 10

Exceptions

An exception is an object that describes an unusual or erroneous situation.

Exceptions

- Exceptions are thrown by a program, and may be caught and handled by another part of the program
- A program can be separated into a normal execution flow and an exception execution flow
- An error is also represented as an object in Java, but usually represents an unrecoverable situation and should not be caught
Exception Handling

- Java has a predefined set of exceptions and errors that can occur during execution.
- A program can deal with an exception in one of three ways:
  - Ignore it
  - Handle it where it occurs
  - Handle it at another place in the program
- The manner in which an exception is processed is an important design consideration.

If an exception is ignored by the program, the program will terminate abnormally and produce an appropriate message. The message includes a *call stack trace* that:
- Indicates the line on which the exception occurred.
- Shows the method call trail that led to the attempted execution of the offending line.
- See Zero.java (page 533)

The Exception Class Hierarchy

- Classes that define exceptions are related by inheritance, forming an exception class hierarchy.
- All error and exception classes are descendents of the `Throwable` class.
- A programmer can define an exception by extending the `Exception` class or one of its descendents.
- The parent class used depends on how the new exception will be used.

### Examples:

Zero.java
Zero_Caught.java

Exceptions Class Hierarchy

- `java.lang.Object`
- `java.lang.Throwable`
- `java.lang.Exception`
- `java.lang.RuntimeException`
- `java.lang.ArithmeticException`
The try Statement

• To handle an exception in a program, the line that throws the exception is executed within a try block
• A try block is followed by one or more catch clauses
• Each catch clause has an associated exception type and is called an exception handler
• When an exception occurs, processing continues at the first catch clause that matches the exception type

The finally Clause

• A try statement can have an optional clause following the catch clauses, designated by the reserved word finally
• The statements in the finally clause always are executed
• If no exception is generated, the statements in the finally clause are executed after the statements in the try block complete
• If an exception is generated, the statements in the finally clause are executed after the statements in the appropriate catch clause complete

Examples:

OutOfBounds.java
OutOfBounds_Caught.java

Exception Hierarchy

• java.lang.Object
• java.lang.Throwable
• java.lang.Exception
• java.lang.RuntimeException
• java.lang.IndexOutOfBoundsException
• java.lang.ArrayIndexOutOfBoundsException

Examples:

NullReference.java
NullReference_Caught.java

Exception Hierarchy

• java.lang.Object
• java.lang.Throwable
• java.lang.Exception
• java.lang.RuntimeException
• java.lang.NullPointerException
Exception Hierarchy

- `java.lang.Object`
- `java.lang.Throwable`
- `java.lang.Exception`
- `java.lang.RuntimeException`
- `java.lang.ClassCastException`

Examples:

- `ClassCast.java`
- `ClassCast_Caught.java`

On-line Java Documentation

- [http://java.sun.com/j2se/1.5.0/docs/api/index.html](http://java.sun.com/j2se/1.5.0/docs/api/index.html)

THE END