Inheritance and Polymorphism: Subclasses and Methods

Jeroo Methods

Method	Purpose	Example
`hop()`	Hop one space ahead. The program terminates with a logic error if the hopping Jeroo lands in the water, lands on another Jeroo, or hops onto a net. A Jeroo can hop onto a flower.	`jessica.hop();`
`hop(number)`	Hop *number* times in a row, where *number* is a positive integer.	`jessica.hop(3);` `jessica.hop(12);`
`pick()`	Pick a flower from the current location. Nothing happens if there is no flower at the current location.	`jessica.pick();`
`plant()`	Plant a flower at the current location. Nothing happens if the jeroo does not have a flower to plant.	`jessica.plant();`
`toss()`	Toss a flower one space ahead. The tossed flower is lost forever. If the flower lands on a net, the net is disabled.	`jessica.toss();`
`turn(relativeDirection)`	Turn in the indicated direction [`turn(AHEAD)` and `turn(HERE)` are meaningless]	`jessica.turn(LEFT);` `jessica.turn(RIGHT);`
`give(relativeDirection)`	Give a flower to a neighboring Jeroo in the indicated direction. Nothing happens if the giving Jeroo has no flowers, or if there is no neighboring Jeroo in the indicated direction. [`give(HERE)` is meaningless]	`jessica.give(LEFT);` `jessica.give(RIGHT);` `jessica.give(AHEAD);`

Directions

Relative Directions	Compass Directions
`LEFT`	`NORTH`
`RIGHT`	`EAST`
`AHEAD`	`SOUTH`
`HERE`	`WEST`

Attributes - values associated with an object

- constructor is responsible to initialize the new object's attributes

Multiple constructors

```java
Jeroo jessica = new Jeroo(8); // specifies 8 flowers
Jeroo jessica = new Jeroo(WEST); // specifies West direction
Jeroo jessica = new Jeroo(WEST, 8); // specifies both direction and flower count
```

Subclasses

extends - allows the subclass to inherit all the attributes and methods of the superclass (parent)

- constructors are not inherited by subclasses

is-a relationship exists between subclass and superclass, since instances of the subclass are also instances of the superclass

```java
public class NewClassName extends ExistingClassName
```

Polymorphism - subclasses can respond to the same method differently

super - calls the constructor of the parent class

overriding methods - change the inherited behavior of a subclass by declaring the same method in the subclass, but with a different function

overloading methods - two methods in the same class that take different parameters to do different behavior

super(params) - modifies the constructor by passing local parameters to superclass' constructor

Algorithm Development Process

1. Obtain a description of the problem
2. Analyze the problem
3. Develop a high-level algorithm
4. Refine the algorithm by adding more detail
5. Review the algorithm

Method Stub - empty method to be filled later