Cheatography

Java_OOD Cheat Sheet Cheat Sheet

by OzzyCodes via cheatography.com/64666/cs/16298/

Loops

for (int i: someArray) {}

while (something) {}

do {something} **while** (true)

Defining Variables

Defining new variable attributes

int *x* = 12;
int *x*; // will be defined as 0

Define by creating new instances

String *x* = new String;

Conditionals

฿ if statement

if (statement) {}

฿ if-else statement

if (statement) {}
else{}

Switch Statement

```
switch (num) {
    case 1: doSomething ();
        break;
    default: doThis ();
        break;
}
```

Override

```
When you have inherit some of the
class from
parents, but you want to do
something different.
In override feature, all the
subclass/class object
will use the newer method.
To make sure JDK knows what you are
doing,
type @Override in front of the
public name. If
the override is unsuccessful, JDK
will returns
```

Override (cont)

```
error.
Example of overriden helloWorld()
method :
Class Student {
    public void helloWorld() {
        System.out.println("Hello")
;
    }
}
Class GradStudent extends Student
    @Override
    public void helloWorld() {
    System.out.println("Hello
World");
    }
}
```

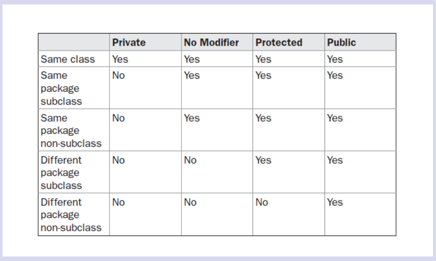
Rules of Overridden methods

```
1. Access modifier priority can
only be
narrower or same as superclass
2. There is the same name method in
superclass / libraries
```

Prime Number Function

```
if (n < 2) {
    return false;
}
for (int i=2; i <= n/i; i++) {
    if (n%i == 0) {
        return false;
    }
    return true;
}
```

Access Modifier



	Private	No Modifier	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	No	Yes	Yes	Yes
Same package non-subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Attribute Modifier

ATTRIBUTE TYPE	ACCESS GRANTED
Private	Allows only in class where variables belong
Public	Allows any class to have this attribute
Protected	The methods or data members declared as protected are accessible within same package or sub classes in different package
Static	Attribute that depends on the class (not object)
Final	Defined once; does not allow any changes/inheritance

java.lang.String

`Find the length` `-> int`

฿ `msg.length()`

`To lower/uppercase` `-> String`

฿ `msg.toLowerCase()`

฿ `msg.toUpperCase()`

`Replace a string` `-> String`

฿ `msg.replaceAll(String a, String b)`

`Split string between delimeter` `-> array`

฿ `msg.split(String delimeter)`

`Start/end with` `-> boolean`

฿ `msg.startsWith(String pre)`

฿ `msg.endsWith(String post)`

`String format` `-> String`

฿ `String.format(String format, Object... args)`

Interface

Interface is different from constructor. It **consists of incomplete assignments**

Interface allows you *to make sure* that any inherited class will implement the methods

(It's like a contract to agree that this thing must be able to do this shit.) The method is then completed in the class that implements it.

Creating a new interface

Constructors

Constructors allow you to create an object template. It consists of **complete procedures**.

Create a blank constructor to allow its extension classes to inherit this *super* constructor.

฿ <modifier> Person () {}

Abstract

Abstract is a type of class but it can consist of **incomplete methods.**

Create new abstract

฿ <access_modifier> abstract class HelloWorld () {}

Interface

`Interface is different from constructor. It`

`consists of incomplete assignments`

`Interface allows you to make sure that any inherited class can do the following methods.`

`The method is then completed in the class that implements it.`

`Creating a new interface`

```
interface Bicycle {
    void speedUp (int increment);
}
----
    class funBike implements Bicycle {
         ...
        void speedUp (int increment) {
            speed += increment;
```

Interface (cont)

```
        }
         ...
}
```

HashList

Methods	Description
void add (int index, Object element)	Add value to a list
Object remove(int index)	Remove item #index from list
Object get(int index)	Retreive item #index from list
void set(int index, Object element)	Set the data to correspond with #index