**How to use “this” keyword**

YouTube tutorial – https://www.youtube.com/watch?v=HVrAEA2sfUU

1st class – apples.java:

**import** java.util.\*;

**class** apples {

**public** **static** **void** main(String args[]) {

anotherclass JustName = **new** anotherclass("Ken");

anotherclass JustAge = **new** anotherclass(19);

anotherclass Both = **new** anotherclass("Ken", 19);

}

}

2nd class – anotherclass.java:

**class** anotherclass {

**int** age;

**int** name;

**public** anotherclass(**int** age){

**this**("Player", age); //By saying “this”, it is giving these

//values to the contructor again (1)

}

**public** anotherclass(String name){

**this**(name, 0);

}

**public** anotherclass(String name, **int** age){

**this**.age = age; //(1) is being redirected here

System.*out*.println(name+" "+**this**.age);

}

}

**Polymorphism and inheritance**

YouTube tutorial - <https://www.youtube.com/watch?v=Lsdaztp3_lw>

3rd class- Animals.java:

**public** **class** Animals {

**private** String name = "Animal";

**public** String favFood = "Food";

**protected** **final** **void** changeName(String newName){

**this**.name = newName;

}

**protected** **final** String getName(){

**return** **this**.name;

}

**public** **void** eatStuff(){

System.*out*.println("Yum "+favFood);

}

**public** **void** walkAround(){

System.*out*.println(**this**.name+" walks around");

}

**public** Animals(){

}

**public** Animals(String name, String favFood){

**this**.changeName(name);

**this**.favFood= favFood;

}

}

2nd class – Cats.java:

**public** **class** Cats **extends** Animals{

**public** String favToy = "Yarn";

**public** **void** playWith(){

System.*out*.println("Yeah "+ favToy);

}

**public** **void** walkAround(){

System.*out*.println(**this**.getName()+" stalks around");

}

**public** String getToy(){

**return** **this**.favToy;

}

**public** Cats(String name, String favFood, String favToy){

**super**(name, favFood);

**this**.favToy=favToy;

}

}

1st class – apples.java:

**class** apples {

**public** **static** **void** main(String args[]) {

Animals genericAnimal = **new** Animals();

System.*out*.println(genericAnimal.getName());

System.*out*.println(genericAnimal.favFood);

Cats morris = **new** Cats("Morris","Tuna","Rubber Mouse");

System.*out*.println();

System.*out*.println(morris.getName());

System.*out*.println(morris.favFood);

System.*out*.println(morris.favToy);

Animals tabby = **new** Cats("Tabby","Salmon","Ball");

*acceptAnimal*(tabby);

}

**public** **static** **void** acceptAnimal(Animals randAnimal){

System.*out*.println();

System.*out*.println(randAnimal.getName());

System.*out*.println(randAnimal.favFood);

System.*out*.println();

randAnimal.walkAround();

Cats tempCat = (Cats) randAnimal;

System.*out*.println(tempCat.favToy);

System.*out*.println(((Cats)randAnimal).favToy);

}

}

The result is:

Animal

Food

Morris

Tuna

Rubber Mouse

Tabby

Salmon

Tabby stalks around

Salmon

Ball

When methods have “final” keyword, they cannot be overwritten.