Baby Objects Q & A

Everything all together all at once

```java
public class Item {
     double price;
     String name;
```
1

```java
public Item(){
  price = 13.45;
  name = "t-shirt";
}
```
2

```java
public Item(double p, String n){
  price = p;
  name = n;
}
```
3

Method Type?

```java
public double getPrice(){
  return price;
}
```
4

```java
public String getName(){
  return name;
}
public String toString(){
  return "The "+name+" costs $"+price;
}
```
5

```java
public void setPrice(double p){
  price = p;
}
```
6

```java
public void setName (String n){
  name = n;
}
```
7

```java
public boolean equals(Item i){
  if(i.getName().equals(name)
        && i.getPrice()==price)
     return true;
  else
     return false;
}
```
8

```java
public int compareTo(Item i){
  //on the basis of price
  if(i.getPrice()>price)
      return -1;
  else if (i.getPrice()==price)
      return 0;
  else
      return 1;
}}
```
9

Instance Variables:

- The variables that you want to store for your object.
- Your object will group these variables together into a new complex type

```
public class Item {
    private double price;
    private String name;
```

Begin the class

Declare the instance variables

First Line of Class File

Class name: ClassName

Instance Variable: double InsVar

```
public ____1____ ____2____ {
```

First Line of Class File

Class name: ClassName

Instance Variable: double InsVar

```
public  class  ClassName {
        ___1___  ___2___
```

First Line of Class File

Class name: Triangle

Instance Variable: int base

```
public ____1____ ____2____ {
```

First Line of Class File

Class name: Triangle

Instance Variable: int base

```
public  class  Triangle  {
```

1

2

First Line of Class File

Class name: Dog

Instance Variable: String name

1

2

```
public ____________ ___________ {
```

First Line of Class File

Class name: Dog

Instance Variable: String name

```
public  class  Dog  {
```

1

2

Constructors

- Initialize and set up memory.
- You need a default AND one with parameters for each instance variable.

```
public Item(){
   price = 13.45;
   name = "t-shirt";
}

public Item(double p, String n){
   price = p;
   name = n;
}
```

Default.
Put a value in each instance variable.

Parameter for each instance variable.

Take each parameter, assign to instance variable.

```java
public Item(){
   price = 13.45;
   name = "t-shirt";
}
```

Constructors are special.

- They have no return type because the type they return is themselves (in this case, an Item).
- They must have the same name as the class.
- When they are called, they are called with the word new and the class name.

```java
Item shoe = new Item(23.45, "flip-flops");
Item shirt = new Item();
```

Default Constructor

Class name: ClassName

Instance Variable: double InsVar

```
          1          2          3          4
public ________ ________(________ ________) {
        return     method     param      param
        type       name       type       name

        5          6
    ________ = ________;
}
```

Default Constructor

Class name: ClassName

Instance Variable: double InsVar

```
public ClassName( ---blank--- ) {
        InsVar = 3.14159;
}
```

1 return type

2 method name

3 param type

4 param name

5

6

Any useful default value

Default Constructor

Class name: Book

Instance Variable: boolean isFiction

```
public ___1___ ___2___(___3___ ___4___) {
    ___5___ = ___6___;
}
```

1 return type

2 method name

3 param type

4 param name

Default Constructor

Class name: Book

Instance Variable: boolean isFiction

```
public Book ( ---blank--- ) {
    isFiction = false ;
}
```

1 return type
2 method name
3 param type
4 param name
5 isFiction
6 false

Any useful default value

Default Constructor

Class name: Person

Instance Variable: char firstInitial

```
public ___1___ ___2___(___3___ ___4___) {
    ___5___ = ___6___;
}
```

1 return type

2 method name

3 param type

4 param name

Default Constructor

Class name: Person

Instance Variable: char firstInitial

```
public  Person  ( ---blank--- ) {
```

1 return type · 2 method name · 3 param type · 4 param name

```
firstInitial = 'a';
}
```

5 · 6

Any useful default value

Customized Constructor

Class name: ClassName

Instance Variable: double InsVar

```
public    1          2        (   3         4      ) {
       ________  ________   ________  ________
        return    method      param     param
         type      name        type      name

      5           6
   ________ = ________;
}
```

Customized Constructor

Class name: ClassName

Instance Variable: double InsVar

```
public ClassName ( double I ) {
    InsVar = I ;
}
```

1 return type

2 method name

3 param type

4 param name

First letter of instance variable

5

6

parameter

Customized Constructor

Class name: Circle

Instance Variable: int radius

```
public __1__ __2__(__3__ __4__) {
    __5__ = __6__;
}
```

1 return type

2 method name

3 param type

4 param name

5

6

Customized Constructor

Class name: Circle

Instance Variable: int radius

```
public Circle ( int r ) {
    radius = r ;
}
```

1 return type

2 method name

3 param type

4 param name

First letter of instance variable

5

6

parameter

Customized Constructor

Class name: Horse

Instance Variable: String name

1 2 3 4

```
public ________ ________(________ ________) {
```

return type | method name | param type | param name

5 6

```
    ________ = ________;
}
```

Customized Constructor

Class name: Horse

Instance Variable: String name

```
public Horse(String n) {
    name = n;
}
```

1 return type

2 method name

3 param type

4 param name

First letter of instance variable

5

6

parameter

Mutators

- Change memory
- You need one for each instance variable.

```
public void setPrice(double p){
   price = p;
}
public void setName (String n){
   name = n;
}
```

Parameter for the instance variable.

Take parameter, assign to the right instance variable.

Mutator

Class name:
ClassName

Instance Variable:
double InsVar

```
           1          2           3          4
public ________  ________(________  ________) {
        return    method      param      param
        type      name        type       name

         5            6
     ________  =  ________;
}
```

Mutator

Class name: ClassName

Instance Variable: double InsVar

```
public void setInsVar(double I){
    InsVar = I;
}
```

1 return type

2 method name

3 param type

4 param name

First letter of instance variable

5

6

parameter

Mutator

Class name: Turtle

Instance Variable: String species

```
public ___1___ ___2___(___3___ ___4___) {
    ___5___ = ___6___;
}
```

1 return type

2 method name

3 param type

4 param name

Mutator

Class name: Turtle

Instance Variable: String species

```
public void setSpecies(String s){
    species = s;
}
```

1 return type

2 method name

3 param type

4 param name

First letter of instance variable

5

6

parameter

Mutator

Class name:
cube

Instance Variable:
int length

```
public ___1___ ___2___(___3___ ___4___) {
    ___5___ = ___6___;
}
```

1 return type

2 method name

3 param type

4 param name

Mutator

Class name: cube

Instance Variable: int length

```
public void setLength( int L ){
    length = L;
}
```

1 return type

2 method name

3 param type

4 param name

First letter of instance variable

5

6

parameter

Accessors

- Access what is stored in memory.
- You need one for each instance variable.

```
public double getPrice(){
   return price;
}
public String getName(){
   return name;
}
public String toString(){
   return "The "+name+" costs $"+price;
}
```

Return type matches the instance variable type.

Return correct instance variable

Make a sentence out of the variables.

Accessor

Class name: ClassName

Instance Variable: double InsVar

```
        1          2          3          4
public _________ _________(_________ _________) {
            return     method     param      param
            type       name       type       name
        5          6
    _________ _________;
}
```

Accessor

Class name:
ClassName

Instance Variable:
double InsVar

```
public double getInsVar(--blank--){
    return InsVar;
}
```

1 return type

Type of instance variable

2 method name

3 param type

4 param name

5

Accessor

Class name:
Fruit

Instance Variable:
char colour

```
public ___1___ ___2___(___3___ ___4___) {
    ___5___ ___6___;
}
```

1 return type

2 method name

3 param type

4 param name

Accessor

Class name: Fruit

Instance Variable: char colour

```
public char getColour( --blank-- ){
    return colour;
}
```

1 return type

Type of instance variable

2 method name

3 param type

4 param name

5

Accessor

Class name: Animal

Instance Variable: int age

```
public __1__ __2__(__3__ __4__) {
    __5__ __6__;
}
```

1 return type

2 method name

3 param type

4 param name

Accessor

Class name:
Animal

Instance Variable:
int age

```
public int getAge(--blank--){
    return age;
}
```

1 return type

Type of instance variable

2 method name

3 param type

4 param name

5

ToString

Class name:
ClassName

Instance Variable:
double InsVar

```
              1            2             3            4
public  __________  __________(__________  __________) {
          return      method       param        param
           type        name        type         name

         5                  6
     __________  ______________________________;
}
```

ToString

Class name:
ClassName

Instance Variable:
double InsVar

```
public String toString(--blank--){
    return “Value is:”+InsVar;
}
```

1 return type

2 method name

3 param type

4 param name

5

6

String with words and instance variables

ToString

Class name:
Tree

Instance Variable:
double height

```
public   1          2        (   3          4       ) {
       ________  ________     ________  ________
       return    method       param     param
       type      name         type      name

      5                   6
  ________  ______________________________;
}
```

ToString

Class name: Tree

Instance Variable: double height

```
public String toString( --blank-- ){
    return "Value is:"+ height;
}
```

1 return type

2 method name

3 param type

4 param name

5

6

String with words and instance variables

ToString

Class name:
House

Instance Variable:
String postalCode

```
public ___1___ ___2___(___3___ ___4___) {
    ___5___ ___________6___________;
}
```

1 return type

2 method name

3 param type

4 param name

ToString

Class name: House

Instance Variable: String postalCode

```
public String toString(--blank--){
    return "Value is:"+postalCode;
}
```

1 return type

2 method name

3 param type

4 param name

5

6

String with words and instance variables

Facilitator: Equals

- Sees if two of your new type are equal

Pass in an object that is the same type as your class

```
public boolean equals(Item i){
   if(i.getName().equals(name)
          && i.getPrice()==price)
      return true;
   else
      return false;
}
```

For each instance variable, see if it matches the parameters' value

Return true if all instance variables match, false otherwise.

Equals

Class name:
ClassName

Instance Variable:
double InsVar

```
public ____1____ ____2____(___3___  ___4___) {
       return type  method name  param type  param name
    if (_____5_____==__6__.____7____)
        return ____8____;
    else
        return ____9____;
}
```

Equals

Class name:
ClassName

Instance Variable:
double InsVar

```
public boolean equals (ClassName C) {
    if (InsVar == C.getInsVar())
        return true;
    else
        return false;
}
```

1 return type
2 method name
3 param type
4 param name
5
6
7
8
9

Equals

Class name: Matrix

Instance Variable: int rows

```
public ____1____ ____2____(____3____ ____4____) {
       return type  method name  param type  param name
    if (____5____==____6____.____7____)
        return ____8____;
    else
        return ____9____;
}
```

Equals

Class name:
Matrix

Instance Variable:
int rows

```
public boolean equals ( Matrix M ) {
    if ( row == M . getRows() )
        return true ;
    else
        return false ;
}
```

1 return type
2 method name
3 param type
4 param name
5
6
7
8
9

Equals

Class name: Equation

Instance Variable: double slope

```
public ____1____ ____2____(____3____ ____4____) {
       return type method name param type param name
    if (_____5_____==__6__.____7____)
        return ____8____;
    else
        return ____9____;
}
```

Equals

Class name: Equation

Instance Variable: double slope

```
public boolean equals(Equation E) {
    if (slope == E.getSlope())
        return true;
    else
        return false;
}
```

1 return type

2 method name

3 param type

4 param name

5

6

7

8

9

Facilitator: CompareTo

- Sees how two of your new type compare, for sorting

```
public int compareTo(Item i){
  //on the basis of price
  if(price>i.getPrice())
      return 1;
  else if (i.getPrice()==price)
      return 0;
  else
      return -1;
}}
```

If the parameter is bigger, return -1

If they are the same, return 0

If the parameter is smaller, return 1

CompareTo

Class name:
ClassName

Instance Variable:
double InsVar

```
public ____1____   ____2____  (____3____   ____4____) {
        return type    method name   param type    param name
    if (____5____>__6__.____7____)
        return ____8____;
    else if (____9____<__10__.____11____)
        return ____12____;
    else
        return ____13____;
}
```

CompareTo

Class name: ClassName

Instance Variable: double InsVar

```
public int compareTo( ClassName C ) {
```

1 return type · 2 method name · 3 p type · 4 param name

Me > Them

```
if ( InsVar > C . getInsVar() )
```

5 · 6 · 7

```
    return 1 ;
```

8 · I win

Me < Them

```
else if ( InsVar < C . getInsVar() )
```

9 · 10 · 11

```
    return -1 ;
```

12 · They win

Me == Them

```
else
    return 0 ;
}
```

13 · We tie

CompareTo

Class name: Tax

Instance Variable: double percent

```
public ____1____ ____2____(____3____ ____4____) {
       return type  method name  param type  param name
    if (____5____>__6__.____7____)
        return ____8____;
    else if (____9____<__10__.____11____)
        return ____12____;
    else
        return ____13____;
}
```

CompareTo

Class name: Tax

Instance Variable: double percent

```
public int compareTo( Tax T ) {
       1 return type  2 method name  3 p type  4 param name
  if ( percent > T . getPercent() )
       5         6   7
     return 1 ; 8
  else if ( percent < T . getPercent() )
            9         10  11
     return -1 ;
            12
  else
     return 0 ;
            13
}
```

Me > Them

I win

Me < Them

They win

Me == Them

We tie

CompareTo

Class name: Ques

Instance Variable: char ans

```
public ___1___     ___2___     ( ___3___    ___4___ ) {
       return type method name   param type param name
    if (___5___ > ___6___.___7___)
        return ___8___;
    else if (___9___ < ___10___.___11___)
        return ___12___;
    else
        return ___13___;
}
```

CompareTo

Class name:
Ques

Instance Variable:
char ans

```
public int compareTo( Ques Q ) {
   if ( ans > Q.getAns() )
      return 1;
   else if ( ans < Q.getAns() )
      return -1;
   else
      return 0;
}
```

1 return type 2 method name 3 p type 4 param name

5 6 7 8 9 10 11 12 13

Me > Them — I win

Me < Them — They win

Me == Them — We tie