int

string

long

double

float

char

§ Code smells

- Code smells
- Duplicated Code
- Long Method
- Large Class
- Long Parameter List
- Divergent Change
- Shotgun Surgery
- Feature Envy
- Data Clumps
- Primitive Obsession
- Switch Statements
- Parallel Inheritance H

Primitive Obsession

Using int, float, string, etc. in place of
domain objects

Domain Objects

The design model, or design document, with no implementation details:

```
Customer
    name
    email
    address
    ...
```

Domain Objects

The design model, or design document, with no implementation details:

```
Customer
    name        Name < 30 characters
    email       Valid email address
    address     Street # and name
    ...
```

Domain Objects

```
int craftId = itemId + random() + favorPts + 5;
```

Compiler: “Looks good to me!”

Domain Objects

```
int craftId = itemId + random() + favorPts + 5;
```

Compiler: “Looks good to me!”

int is ***not*** a domain object, it’s a ***hardware representation***

Domain Objects

Using primitives:

- Bypass type checking — “If everything is an int...”
- Violate encapsulation — “Types leak & propagate”
- Limit documentation — ”What can I do with it?”

Real World Issues

Mixing Id types

```
if (this.SharedItemLoot.SecondaryItemId > 0 &&
    craftIdChangeEvent.CraftId !=
    this.SharedItemLoot.SecondaryItemId) {
        return false;
}
```

Compiler: “Looks good to me”

Real World Issues

Mixing Id types

```
if (this.SharedItemLoot.SecondaryItemId > 0 &&
    craftIdChangeEvent.CraftId !=
    this.SharedItemLoot.SecondaryItemId) {
        return false;
}
```

Implicit “union” between craftId and itemId using int

Camparing itemId with int vs using ItemId.IsValid().

Real World Issues

Exposing implementation

```
public static Cell GetCellWithCellId(int cellId) {
    ...
}
```

Seems reasonable

Real World Issues

Exposing implementation

```
public static Cell GetCellWithCellId(int cellId) {
    ...
}
```

Change to 64 bit cell Ids: *modify 23 files*

Number Types

Cardinal	Quantity	`cash, part.qty`
Ordinal	Rank/Position	`level, chapter`
Nominal	Identifier	`cellId, itemId`

int, long — Whatever you want to do!

`+ - * / & | << >>`

Number Types

Cardinal	Quantity	100, 55, -27
Ordinal	Rank/Position	1, 2, 3, 4, ...
Nominal	Identifier	1001, 20015

int, long — Whatever you want to do!

```
itemId << cash // ???
```

Back to Domain Objects

```
struct CraftId {
    public int value { get; private set; }
                “has-a”
    CraftId(int value) : this() {
        this.value = value;
    }
}
```

- Can be type checked
- Encapsulates value
- Helps self-document

Back to Domain Objects

```
struct CraftId {
    public int value { get; private set; }

    CraftId(int value) : this() {
        this.value = value; <= Useful breakpoint!
    }
}
```

Value objects (struct) are just as ***efficient*** as primitives

C# allows **operator overloading**: (Cash) + (Cash) = (Cash)

More Examples

```
class Point2D {
    float x; float y;
    ...
}
```

We don’t have any issues with this

More Examples

```
class Point1D {
    float x;
    ...
}
```

Why is this any different? “has-a” vs “is-a”.

Point1D is a domain object, not an hardware type

More Examples

```
void showEmailAddress(string emailAddress) {
    // Do we know if we have a valid email address?
}
```

More Examples

```
class EmailAddress {
   string value; // “has-a”
   EmailAddress(string value) { /* validate */ }
   string GetString() { return value; }
}

void showEmailAddress(EmailAddress emailAddress) {
    // We know we have a valid email address!
}
```

Domain Modelling

Domain Model		Implementation
Quest	=>	class Quest
Cell	=>	class Cell
Email	=>	string or EmailAddr
Craft Id	=>	int or CraftId
Cash	=>	int or Currency.Cash

Domain Modelling

Domain Model		Implementation
Quest	=>	class Quest
Cell	=>	class Cell
Email	=>	~~string or~~ EmailAddr
Craft Id	=>	~~int or~~ CraftId
Cash	=>	~~int or~~ Currency.Cash

Future Topics

- ✓ Keyword Arguments
- ✓ **Primitive Obsession**
- ☐ Avoiding Singletons
- ☐ Collections & LINQ
- ☐ Functional.Maybe

FIN

http://richiban.uk/2015/01/20/the-dangers-of-primitive-obsession

https://sourcemaking.com/refactoring/primitive-obsession