COMP 110/L Lecture 8

Kyle Dewey

Outline

- “Random” numbers
- `if / else if /... /else`

Random Numbers

Random Numbers

Random numbers can be generated with `java.util.Random`

Random Numbers

Random numbers can be generated with `java.util.Random`

```
Random r = new Random();
int isRandom = r.nextInt();
```

(generates any random integer)

Random Numbers

Random numbers can be generated with

`java.util.Random`

```
Random r = new Random();
int isRandom = r.nextInt();
```

(generates any random integer)

```
Random r = new Random();
int isRandom = r.nextInt(10);
```

Random Numbers

Random numbers can be generated with

`java.util.Random`

```
Random r = new Random();
int isRandom = r.nextInt();
```

(generates any random integer)

```
Random r = new Random();
int isRandom = r.nextInt(10);
```

(generates one of the following random integers:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

Example: `RandomExample.java`

How `Random` Works

- Not actually random, but *psuedorandom*
- General idea:
  - Start with a *seed* value
  - Do a computation on it
  - Computation produces a psuedorandom value and a new seed
  - Repeat for infinity

Passing Seed Values

Seeds can be explicitly passed to `Random`

Passing Seed Values

Seeds can be explicitly passed to `Random`

```
Random r = new Random(123);
// seed is 123
int isRandom = r.nextInt();
```

Passing Seed Values

Seeds can be explicitly passed to `Random`

```
Random r = new Random(123);
// seed is 123
int isRandom = r.nextInt();
```

Always produces `-1188957731`

Example:
`RandomExampleWithSeed.java`

Utility of Setting Seeds

Predictable random values mean predictable tests.

Utility of Setting Seeds

Predictable random values mean predictable tests.

```
@Test
public void testRandomCalculation() {
  long seed = 123l;
  assertEquals(Something.calc(seed),
               42);
}
```

Without Explicit Seeds

If no seed is passed, `Random` will generate a seed based off of another source, such as the current time.

Without Explicit Seeds

If no seed is passed, `Random` will generate a seed based off of another source, such as the current time.

```
Random r = new Random();
int isRandom = r.nextInt();
```

if / else if /... /else

`if` / `else`

So far: only two branches allowed

if / else

So far: only two branches allowed

```
if (x > 5) {
  return 7;
} else {
  return 8;
}
```

`if` / `else` With More Than Two Branches

More branches are possible

`if` / `else` With More Than Two Branches

More branches are possible

```
if (x == 0) {
  return 7;
} else if (x < 10) {
  return 8;
} else if (x > 50) {
  return 9;
} else {
  return 10;
}
```

Example:
`IfElseIfElse.java`

Note on Testing

Good idea to have at least one test for each branch

Note on Testing

Good idea to have at least one test for each branch

```
if (x == 0) {
  return 7;
} else if (x < 10) {
  return 8;
} else if (x > 50) {
  return 9;
} else {
  return 10;
}
```

Note on Testing

Good idea to have at least one test for each branch

Good test inputs?

```
if (x == 0) {
  return 7;
} else if (x < 10) {
  return 8;
} else if (x > 50) {
  return 9;
} else {
  return 10;
}
```

Note on Testing

Good idea to have at least one test for each branch

Good test inputs?

```
if (x == 0) {  0
  return 7;
} else if (x < 10) {
  return 8;
} else if (x > 50) {
  return 9;
} else {
  return 10;
}
```

Note on Testing

Good idea to have at least one test for each branch

Good test inputs?

```
if (x == 0) {  0
  return 7;
} else if (x < 10) { 1
  return 8;
} else if (x > 50) {
  return 9;
} else {
  return 10;
}
```

Note on Testing

Good idea to have at least one test for each branch

Good test inputs?

```
if (x == 0) {  0
  return 7;
} else if (x < 10) { 1
  return 8;
} else if (x > 50) { 51
  return 9;
} else {
  return 10;
}
```

Note on Testing

Good idea to have at least one test for each branch

Good test inputs?

```
if (x == 0) {  0
  return 7;
} else if (x < 10) { 1
  return 8;
} else if (x > 50) { 51
  return 9;
} else {  50
  return 10;
}
```

Example:

`IfElseIfElseTest.java`