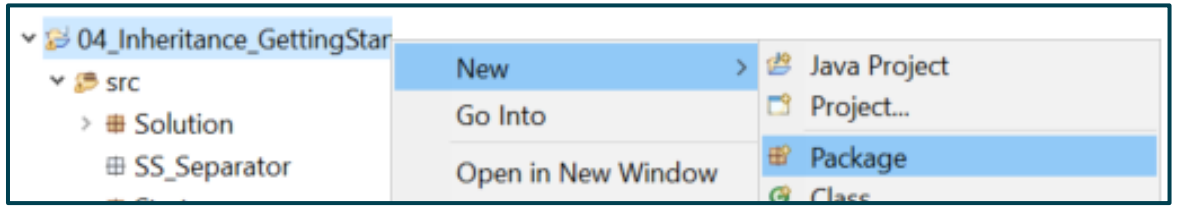
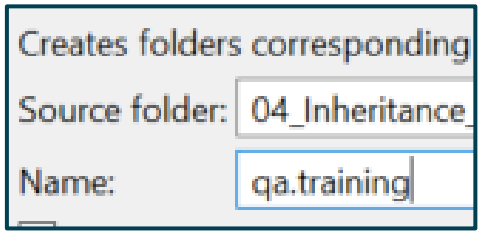
POWERING POTENTIAL

Fundamentals – recap

Java: First few concepts – recap

- Package – used to give a type a unique name






```
package qa.training;

public class Student {
    Trainter trainer;
}
```

```
package qa.training;

public class Trainer {
    Student student;
}
```

Classes in the same namespace can refer to classes

```
package sales;
import qa.training.*;

public class  Program {
   public static void  main( String[] args)
{
            Student student;
```

Classes in the a different package

First few concepts – recap

- **8 primitive types supplied as value-types**
  byte, char, short, `int`, `long`, `float`, `double`, `boolean`

- **Compound operators**
  `+=`, `-=`, `*=`, `/=`, `%=`

```
int x = 4, y = 6;

x = x + y;

x += y;
```

X = 10

X = 16

```
int x = 5;
x = x + 1;
x++;
++x;
```

X = 8

```
int x = 5;
x = x -1;
x--;
--x;
```

X = 5

Pre and post increment/decrement

```
int x = 4, y = 6;

x = y++;
```

X = 6
Y = 7

What are the values of x and Y

```
int x = 4, y = 6;

x = ++y;
```

X = 7
Y = 7

What are the values of x and Y

Methods and parameter passing

```
public class HR {
        public boolean register(String name, int age, String courseName) {
                return true;
        }
}
```

```
public class Student {
    private String name;
    int age;
    public void attendCourse() {
         HR hr = new HR();
         boolean success = hr.register(name,age,"Java");
    }
}
```

Methods and parameter passing

```
public class HR {
        public boolean register(String name, int age, String courseName) {
                return true;
        }

        public boolean register(Student student, String courseName) {
                return true;
        }
}
```

```
public class Student {
    private String name;
    int age;
    public void attendCourse() {
         HR hr = new HR();
         boolean success = hr.register(this,"Java");
    }
}
```

Formatted output via System,out.printf()

- Java:

```
System.out.println("Name: %s, Age: %d", name, age);
```

```
String result = String.format("Name: %s, Age: %d", name, age);
System.out.println(result);
```

Conditionals and operators

```
if(age > 16) {

}
```

```
if (age < 16 || age > 18) {

}
```

```
if(age > 16) {

}
else {

}
```

```
switch (age) {
        case 15:
                // do something
                break;
        case 16:
        case 17:
                // do something
                break;
        default:
                // do something
                break;
}
```

```
if(age > 30) {

}
else if(age > 18) {

}
else {

}
```

Reading data in from console

- There is no `System.in.readln()`. Use `java.util.Scanner`

```
Scanner s = new Scanner(System.in);

System.out.println("What is your name?");
String name = s.nextLine();

System.out.println("What is your age?");
int age = s.nextInt();
```

Arrays – simplest sort of collection

```
int[] votes;                                  // declare
votes = new int[3];                           // create

String[] names = new String[6];               // declare & create

int[] numbers = {4,3,8,12};                   // declare & create & fill


names[0] = "Bob";
votes[0] = 80;
```

Arrays

```
int[] numbers = {4, 3, 8, 12, 45, 5};
```

```
int len = numbers.length;
int i = 0;
while (i < len) {
        print(numbers[i++]);
}
```

```
for (int i=0; i<numbers.length; i++)
        print(numbers[i]);
```

```
for (int x : numbers)
        print(x);
```

Defining your own types

```java
public enum Status{
    Active,
    Completed,
    Left
}
```

```java
public class Student {
    private String name, course;
    private int age;
    private Status status;

    public Status getStatus() {
        return status;
    }
    public void setStatus(Status status) {
        this.status = status;
    }
    // other getters and setters
}
```

```java
Student stu = new Student();
stu.setStatus(Status.Active);
stu.setName("Bob");
stu.setAge(25);
// Set other fields
```

Defining your own types - Constructors

```java
public class Student {
    private String name, course;
    private int age;
    private Status status;

    public Student(String name, int age, String courser, Status status) {
        this.name = name;
        this.age = age;
        this.course = course;
        this.setStatus(status);
    }
    // the other getters and setters
}
```

```java
Student stu = new Student();
```

```java
Student stu = new Student("Bob",25,"Java",Status.Active);
```

String Class

Strings are immutable, no methods to change their state

- Methods like `trim()`, `toUpperCase()` etc. all return new String ref

```
String name = “Fred”;
char c = name.charAt(2);
```

- In Java, chars are NOT enumerable without invoking `toCharArray()`

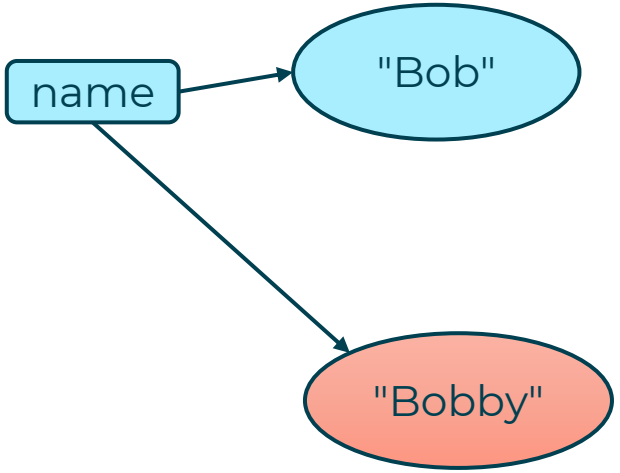
```
for (char c : name.toCharArray()) {
      ...
```

Strings are immutable

```
name = "Bob";
```

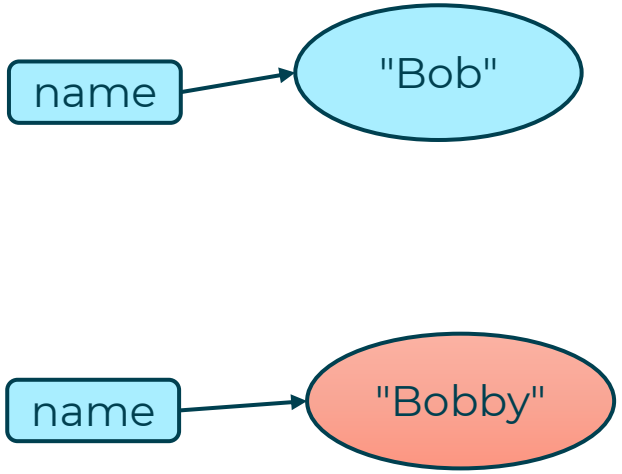
```
name[0] ='R';
```

```
name = name + "by";
```




```
public static void main(String[] args) {
    String name = "Bob";
    changeName(name);
}

private static void changeName(String name) {
    name = "Bobby";
}
```



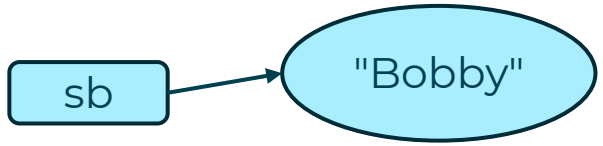
StringBuilder

`StringBuilder` is a mutable String buffer

- Key methods: `append()`, `insert()`, `replace()`, `delete()`

```
StringBuilder sb = new StringBuilder("Bob");


sb.append("by");
```




```
String name = sb.toString();
```

Generic Collection classes

- `ArrayList<E>, ArrayDeque<E>, TreeMap<K,V>`

```
ArrayList<String> names = new ArrayList<String>();
names.add("Bob");
names.add("Linda");

print(names.get(0));

for (String x : names)
        System.out.println(x);
```

FILE I-O

class `File`

- Provides instance methods to allow manipulation of files and directories

Java example – Copy using stream classes

```
public static void copy (String inFile, String outFile)
                                        throws IOException {
  byte[] bytes = new byte[128];

  FileInputStream fis  = new FileInputStream(inFile);
  FileOutputStream fos = new FileOutputStream(outFile);

  int count = 0, read = 0;

  while ((read = fis.read(bytes)) != -1) {
    fos.write(b, 0, read); // mainly 128 at a time
    count += read;
  }
  System.out.printf("Wrote: %d bytes\n “, count);
  fis.close();
  fos.close();
}
```