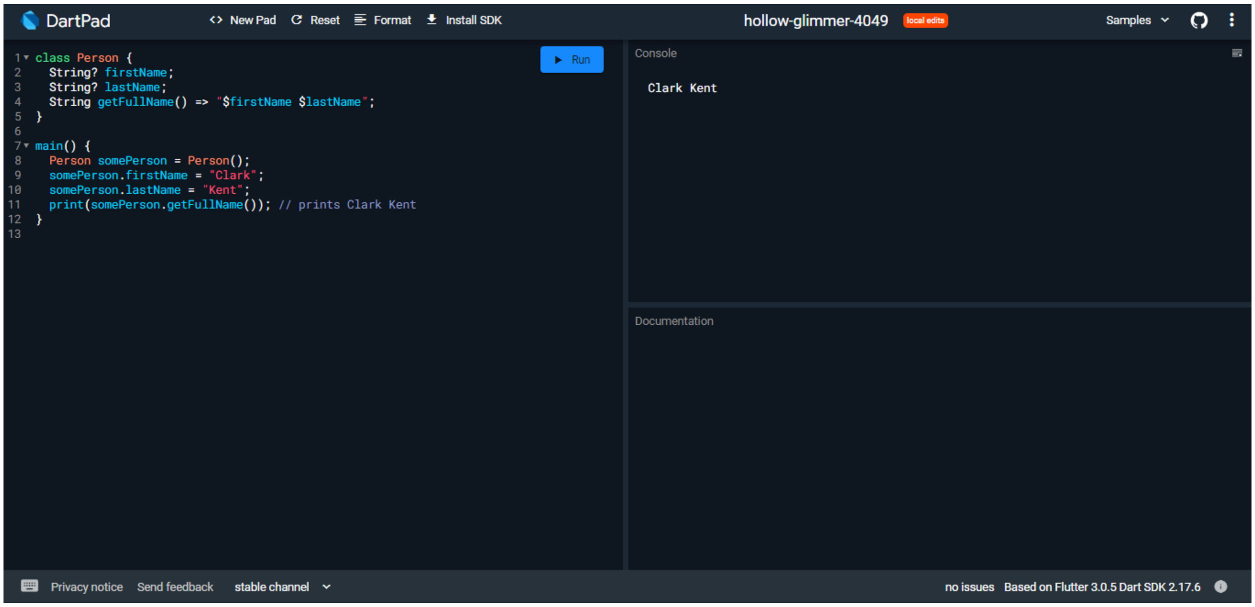
Nama : Dhika Ainul Luthfi

Kelas : TSA MOBILE

Struktur Kelas

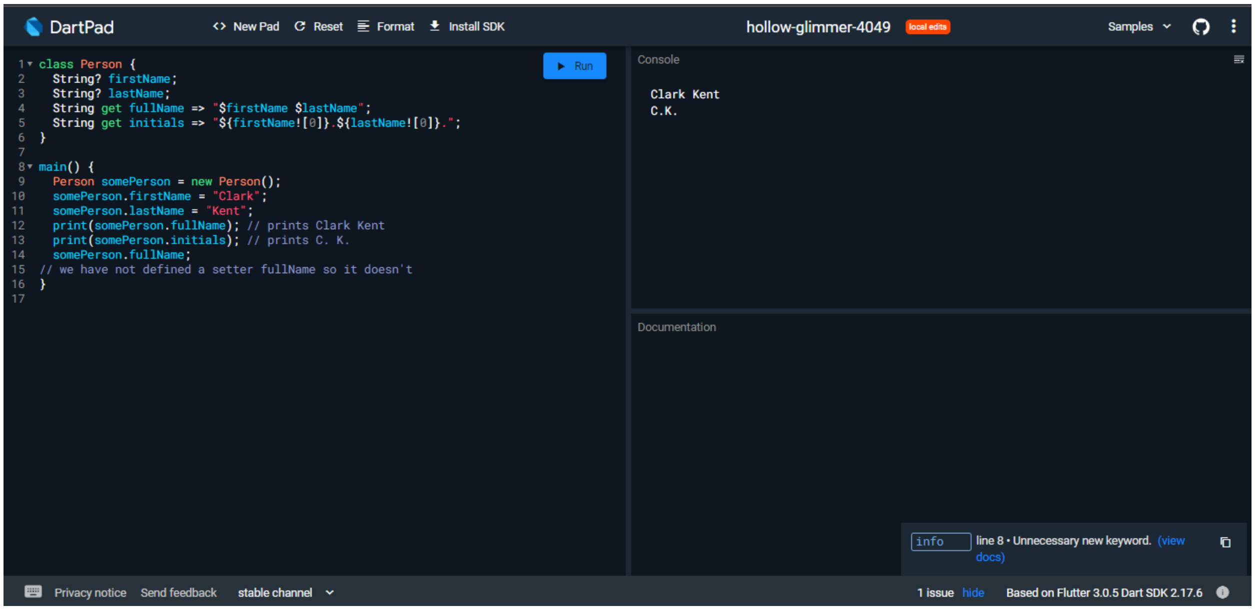


```dart
class Person {
  String? firstName;
  String? lastName;
  String getFullName() => "$firstName $lastName";
}

main() {
  Person somePerson = Person();
  somePerson.firstName = "Clark";
  somePerson.lastName = "Kent";
  print(somePerson.getFullName()); // prints Clark Kent
}
```

```
Clark Kent
```

Kode Setter dan Getter

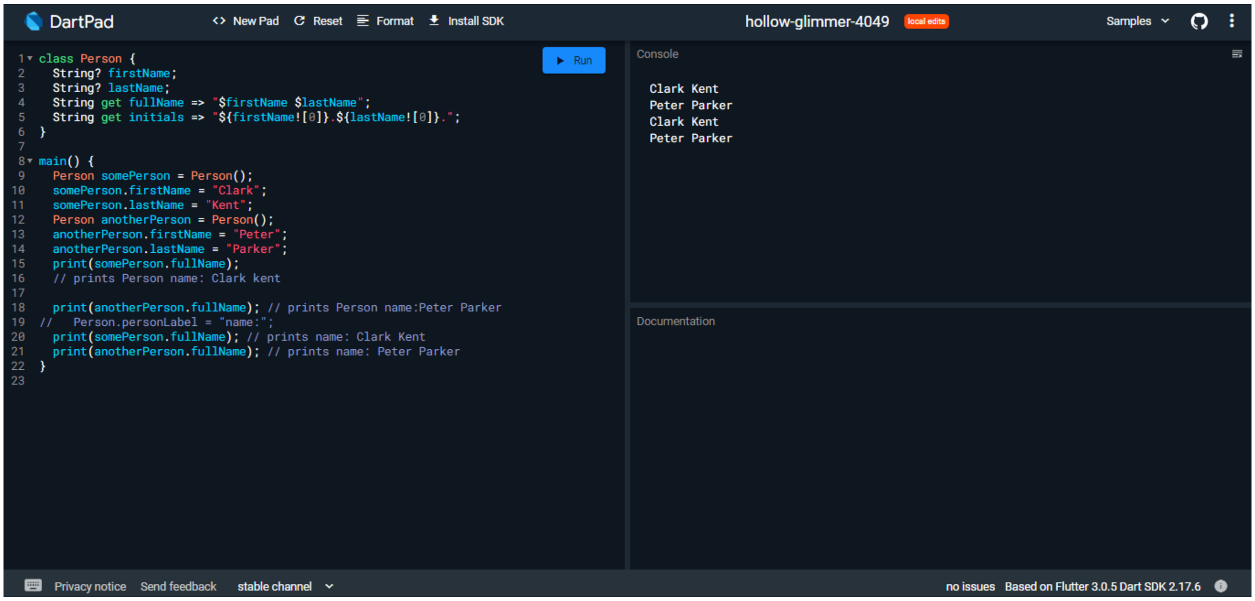


```dart
1  class Person {
2    String? firstName;
3    String? lastName;
4    String get fullName => "$firstName $lastName";
5    String get initials => "${firstName![0]}.${lastName![0]}.";
6  }
7
8  main() {
9    Person somePerson = new Person();
10   somePerson.firstName = "Clark";
11   somePerson.lastName = "Kent";
12   print(somePerson.fullName); // prints Clark Kent
13   print(somePerson.initials); // prints C. K.
14   somePerson.fullName;
15 // we have not defined a setter fullName so it doesn't
16 }
17
```

```
Clark Kent
C.K.
```

info line 8 • Unnecessary new keyword. (view docs)

Static Fields and Methods



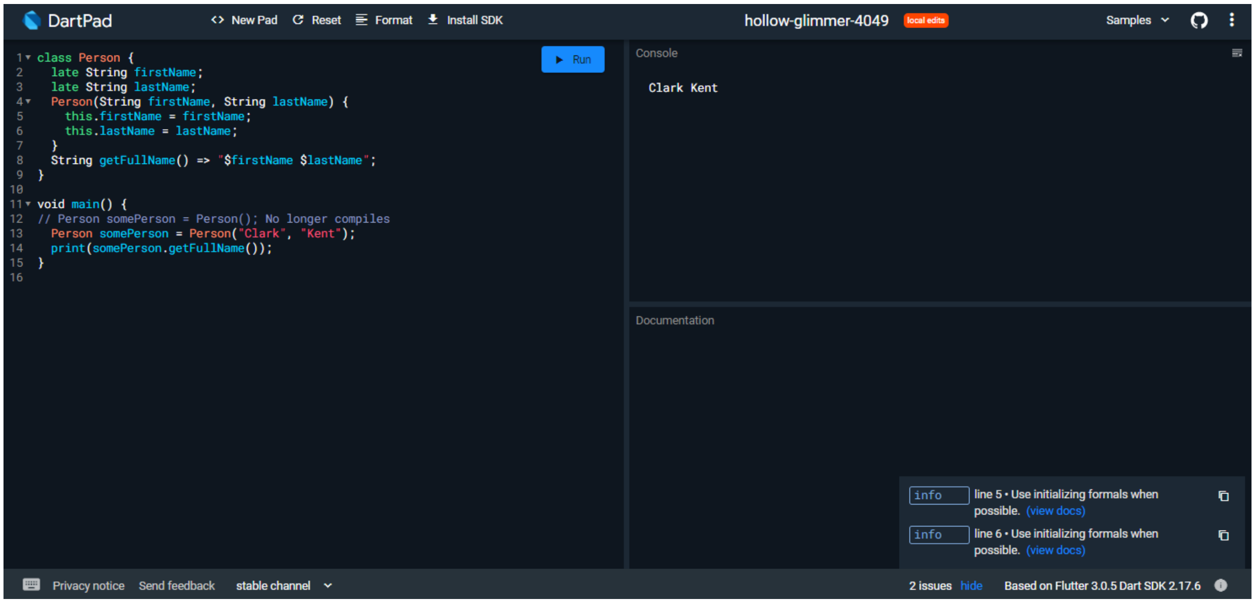
```dart
class Person {
  String? firstName;
  String? lastName;
  String get fullName => "$firstName $lastName";
  String get initials => "${firstName![0]}.${lastName![0]}.";
}

main() {
  Person somePerson = Person();
  somePerson.firstName = "Clark";
  somePerson.lastName = "Kent";
  Person anotherPerson = Person();
  anotherPerson.firstName = "Peter";
  anotherPerson.lastName = "Parker";
  print(somePerson.fullName);
  // prints Person name: Clark kent

  print(anotherPerson.fullName); // prints Person name:Peter Parker
//   Person.personLabel = "name:";
  print(somePerson.fullName); // prints name: Clark Kent
  print(anotherPerson.fullName); // prints name: Peter Parker
}
```

```
Clark Kent
Peter Parker
Clark Kent
Peter Parker
```

Konstruktor



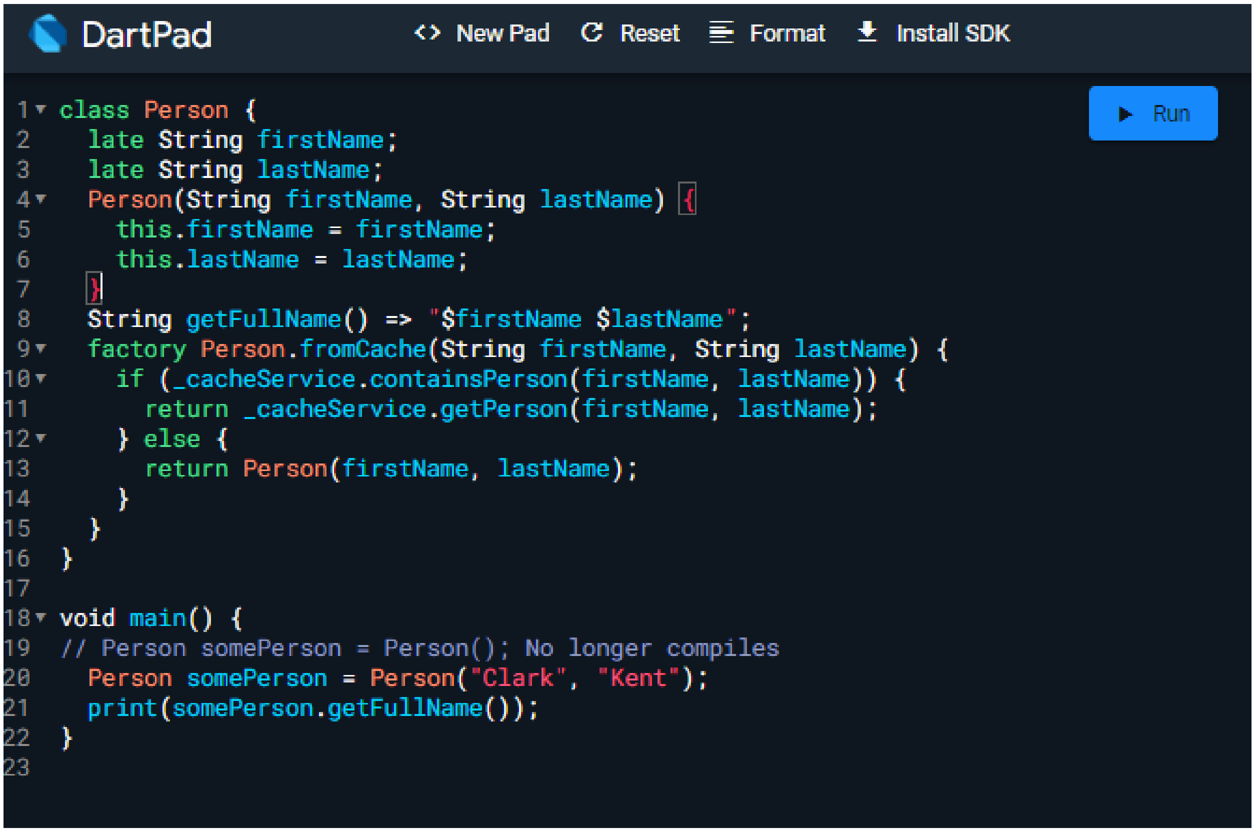
```dart
1  class Person {
2    late String firstName;
3    late String lastName;
4    Person(String firstName, String lastName) {
5      this.firstName = firstName;
6      this.lastName = lastName;
7    }
8    String getFullName() => "$firstName $lastName";
9  }
10
11 void main() {
12 // Person somePerson = Person(); No longer compiles
13   Person somePerson = Person("Clark", "Kent");
14   print(somePerson.getFullName());
15 }
16
```

```
Clark Kent
```

info line 5 • Use initializing formals when possible. (view docs)

info line 6 • Use initializing formals when possible. (view docs)

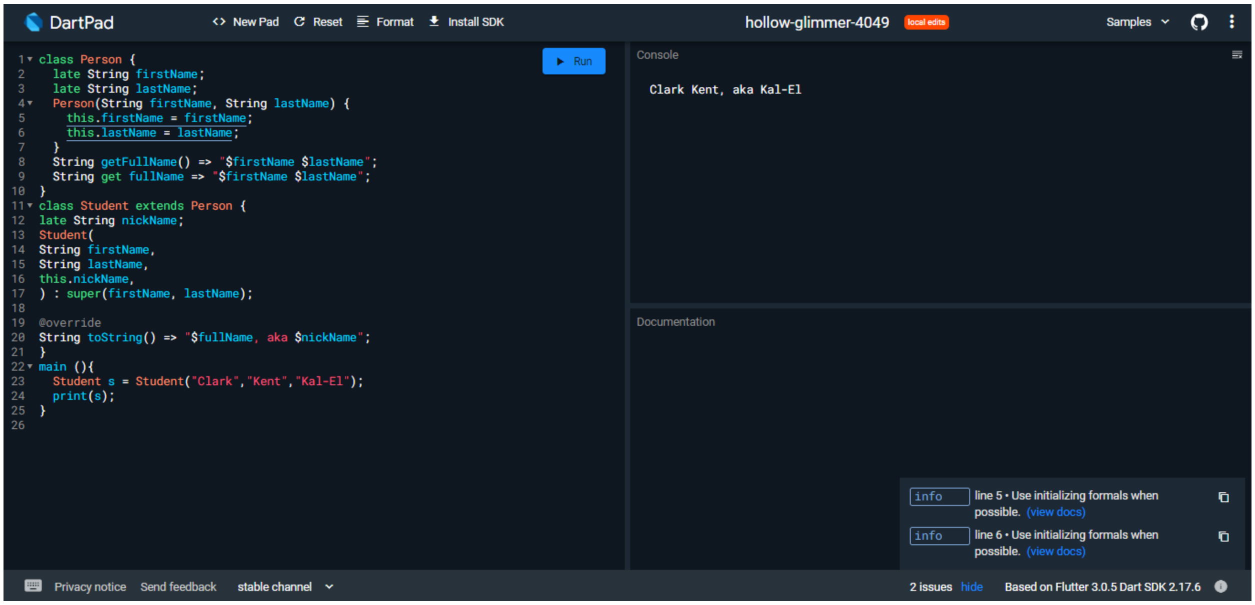
Factory Constructor



```dart
class Person {
  late String firstName;
  late String lastName;
  Person(String firstName, String lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }
  String getFullName() => "$firstName $lastName";
  factory Person.fromCache(String firstName, String lastName) {
    if (_cacheService.containsPerson(firstName, lastName)) {
      return _cacheService.getPerson(firstName, lastName);
    } else {
      return Person(firstName, lastName);
    }
  }
}

void main() {
// Person somePerson = Person(); No longer compiles
  Person somePerson = Person("Clark", "Kent");
  print(somePerson.getFullName());
}
```

Class Inheritance



```dart
1  class Person {
2    late String firstName;
3    late String lastName;
4    Person(String firstName, String lastName) {
5      this.firstName = firstName;
6      this.lastName = lastName;
7    }
8    String getFullName() => "$firstName $lastName";
9    String get fullName => "$firstName $lastName";
10 }
11 class Student extends Person {
12 late String nickName;
13 Student(
14 String firstName,
15 String lastName,
16 this.nickName,
17 ) : super(firstName, lastName);
18
19 @override
20 String toString() => "$fullName, aka $nickName";
21 }
22 main (){
23   Student s = Student("Clark","Kent","Kal-El");
24   print(s);
25 }
26
```

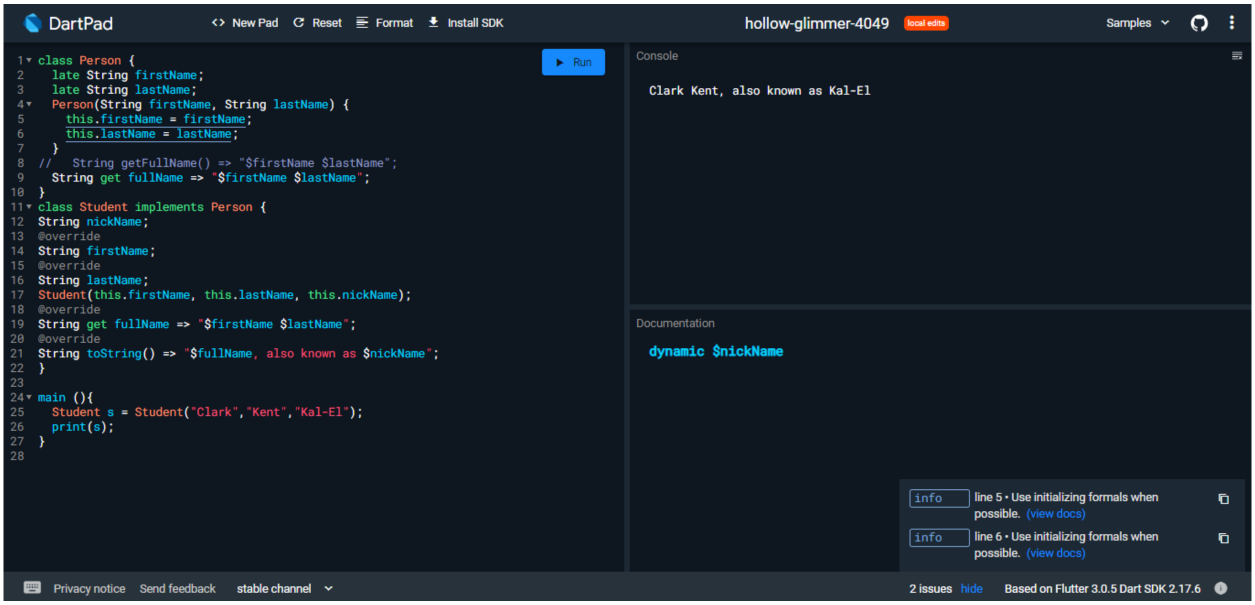
```
Clark Kent, aka Kal-El
```

info line 5 • Use initializing formals when possible. (view docs)

info line 6 • Use initializing formals when possible. (view docs)

2 issues hide Based on Flutter 3.0.5 Dart SDK 2.17.6

Interfaces



```
1  class Person {
2    late String firstName;
3    late String lastName;
4    Person(String firstName, String lastName) {
5      this.firstName = firstName;
6      this.lastName = lastName;
7    }
8  //   String getFullName() => "$firstName $lastName";
9    String get fullName => "$firstName $lastName";
10 }
11 class Student implements Person {
12 String nickName;
13 @override
14 String firstName;
15 @override
16 String lastName;
17 Student(this.firstName, this.lastName, this.nickName);
18 @override
19 String get fullName => "$firstName $lastName";
20 @override
21 String toString() => "$fullName, also known as $nickName";
22 }
23
24 main (){
25   Student s = Student("Clark","Kent","Kal-El");
26   print(s);
27 }
28
```

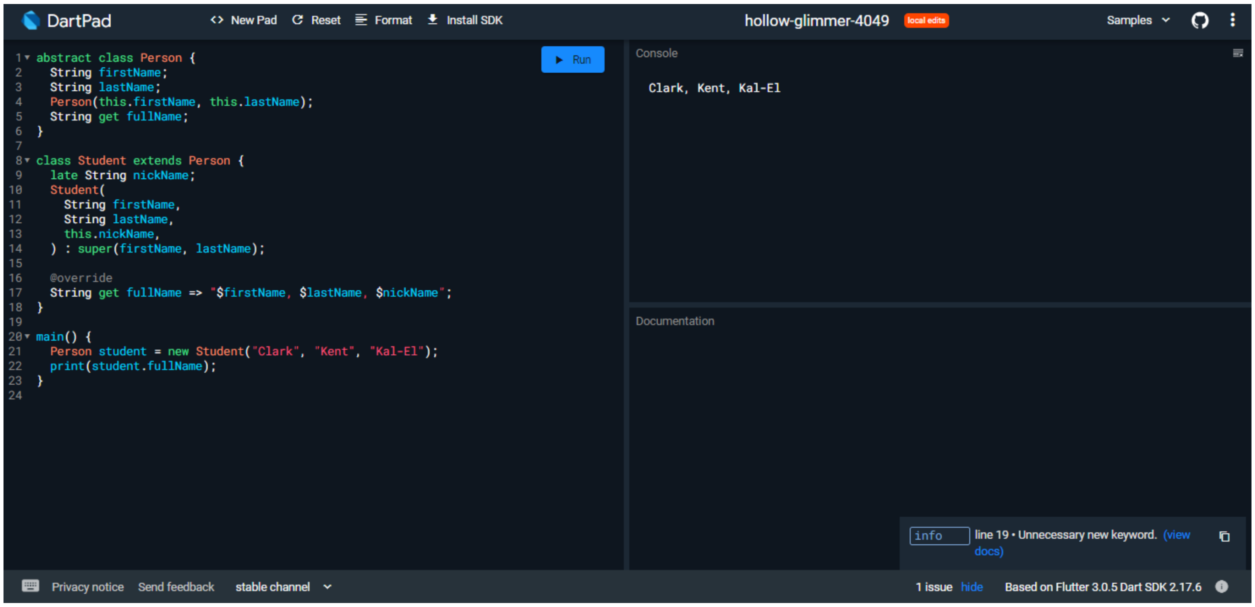
```
Clark Kent, also known as Kal-El
```

dynamic $nickName

info line 5 • Use initializing formals when possible. (view docs)

info line 6 • Use initializing formals when possible. (view docs)

Abstract

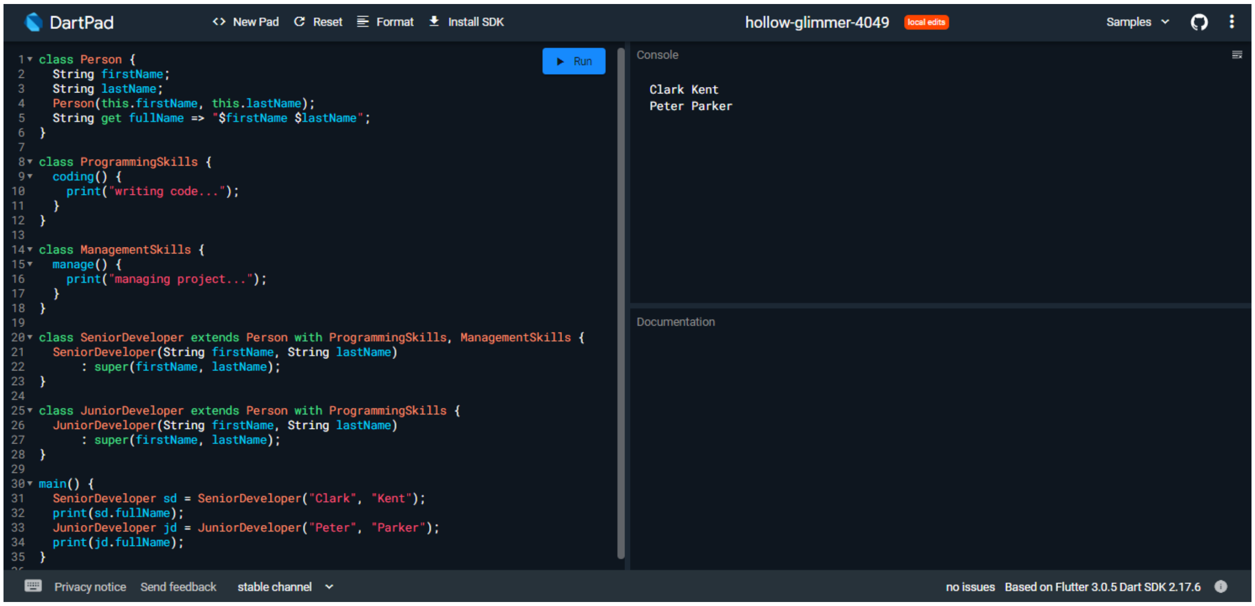


```
1  abstract class Person {
2    String firstName;
3    String lastName;
4    Person(this.firstName, this.lastName);
5    String get fullName;
6  }
7
8  class Student extends Person {
9    late String nickName;
10   Student(
11     String firstName,
12     String lastName,
13     this.nickName,
14   ) : super(firstName, lastName);
15
16   @override
17   String get fullName => "$firstName, $lastName, $nickName";
18 }
19
20 main() {
21   Person student = new Student("Clark", "Kent", "Kal-El");
22   print(student.fullName);
23 }
24
```

```
Clark, Kent, Kal-El
```

info line 19 • Unnecessary new keyword. (view docs)

Mixins



```
class Person {
  String firstName;
  String lastName;
  Person(this.firstName, this.lastName);
  String get fullName => "$firstName $lastName";
}

class ProgrammingSkills {
  coding() {
    print("writing code...");
  }
}

class ManagementSkills {
  manage() {
    print("managing project...");
  }
}

class SeniorDeveloper extends Person with ProgrammingSkills, ManagementSkills {
  SeniorDeveloper(String firstName, String lastName)
      : super(firstName, lastName);
}

class JuniorDeveloper extends Person with ProgrammingSkills {
  JuniorDeveloper(String firstName, String lastName)
      : super(firstName, lastName);
}

main() {
  SeniorDeveloper sd = SeniorDeveloper("Clark", "Kent");
  print(sd.fullName);
  JuniorDeveloper jd = JuniorDeveloper("Peter", "Parker");
  print(jd.fullName);
}
```

```
Clark Kent
Peter Parker
```