Intellij IDEA (autore: Vittorio Albertoni)

Premessa

La prima versione di Intellij Idea è stata rilasciata, nel gennaio del 2001, a Praga ad opera della Intellij, poi divenuta JetBrains, azienda di software che ha tuttora la sede principale a Praga.

Nasce come IDE (Integrated Develpment Environment) per il linguaggio di programmazione Java, nella stessa località, Praga, dove quattro anni prima era nato Net Beans, il decano degli IDE per Java e appena prima dell'altro famoso IDE per Java, Eclipse.

Col tempo si arricchisce sempre più fino a diventare probabilmente il più ricco IDE, a supporto di un cinquantina di linguaggi, tecnologie e framework di programmazione.

Viene distribuito in due versioni: la versione Ultimate, la più completa, a pagamento e sotto licenza proprietaria e la versione Community, per Java Virtual Machine e Android, gratuita e sotto licenza libera Apache2.

La versione Community è stata adottata da Google alla base di Android Studio, l'ambiente di sviluppo integrato per applicazioni Android.

Con la versione Community, presente sul computer anche l'Android Software Development Kit (Android SDK) si possono fare le stesse cose che si fanno con Android Studio.

Esiste anche una versione Edu con un supporto speciale per studenti ed educatori.

In questo manuale tratto la programmazione per la Java Virtual Machine (linguaggi Java e Kotlin) con la versione Community ma tutto ciò che dirò vale anche per la versione Edu, che, in più della versione Community, offre una serie di suggerimenti didattici, a scelta per allievi e per insegnanti.

Basta comunque installare un plugin per arricchire la versione Community e farla diventare uguale alla versione Edu.

La versione di riferimento è la 2021.3 rilasciata nel novembre 2021, la più aggiornata disponibile nel momento in cui scrivo (gennaio 2022).

Indice

1 Installazione

Prima di installare Intellij IDEA è bene che ci accertiamo di avere installati i kit di sviluppo per i programmi che vogliamo produrre. Per programmare in linguaggio Java il JDK (Java Development Toolkit) e per programmare in linguaggio Kotlin il kotlin compiler.

Intellij IDEA saprà ritrovarli al momento dell'installazione e predisporsi per utilizzarli.

Intellij IDEA si trova all'indirizzo *https://www.jetbrains.com/idea/*.

La home page ne illustra i pregi e cliccando in alto a destra sul pulsante DOWNLOAD apriamo la pagina da cui scaricare i file per l'installazione.

Intellij IDEA è disponibile per Windows, Mac OS e Linux; la pagina si apre riconoscendo il sistema operativo su cui stiamo operando e ci propone file installatori `.exe` se siamo su Windows, file installatori `.dmg` se siamo su Mac o tar ball `.tar.gz` se siamo su Linux.

Ci vengono proposte le versioni Ultimate (prova gratuita per 30 giorni), Community (gratuita) e Edu (gratuita).

Per le nostre esigenze basta e avanza la versione Community.

Sistema Windows

Basta lanciare l'eseguibile che abbiamo scaricato e seguire le istruzioni.

Sistema Mac OS X

Si monta l'immagine scaricata e si copia Intellij IDEA nel folder delle Applicazioni.

Sistema Linux

Come super utente si estrae il tar ball in un luogo adatto (è consigliabile `/opt`).

Nella sottodirectory `/bin` della directory che si è creata troviamo il comando `idea.sh` che avvia il programma. Se non si crea un lanciatore automaticamente possiamo crearlo noi (nella directory di installazione si trova l'icona che possiamo utilizzare per marchiare il lanciatore).

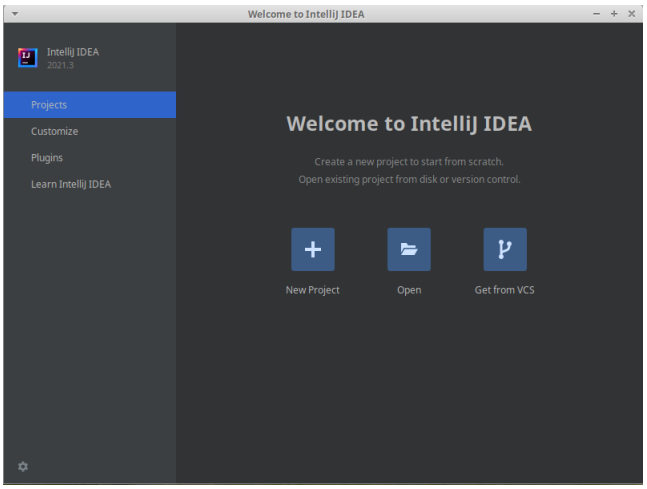
Chi lavora su Ubuntu, su una sua derivata o su altro sistema Linux che supporti snap può installare Intellij IDEA Community con il comando a terminale

```
sudo snap install intellij-idea-community --classic
```

2 Configurazione

Al primo lancio del programma ci viene richiesto se Intellij IDEA debba recuperare configurazioni di precedenti installazioni e rispondiamo di conseguenza.

Appare così la schermata iniziale



Se non abbiamo recuperato configurazioni di precedenti installazioni possiamo creare ora una configurazione, a partire dall'aspetto.

Allo scopo scegliamo sulla sinistra della finestra la modalità CUSTOMIZE e si apre la relativa finestra.

Per default Intellij IDEA si presenta a sfondo scuro, con l'impostazione denominata Darcula.

Se più ci aggrada una impostazione a sfondo chiaro, ed è il caso mio, nella nuova finestra che si è aperta, nella zona COLOR THEME, per scorrimento nella finestrella, sostituiamo Darcula con Intellij Light.

Un'abitudine che Intellij IDEA ha per default è quella di aprirsi ogni volta sull'ultimo progetto su cui si è lavorato. Evidentemente ciò viene bene per chi lavora settimane o mesi su un impegnativo solo progetto in quanto è sollevato dal caricarlo ogni volta.

Per principianti come noi, che forse esauriamo i nostri esercizi in una sola seduta, questa abitudine può risultare fastidiosa.

Per levarla, scegliamo, in fondo alla finestra di configurazione, la voce ALL SETTINGS, successivamente la voce APPEARANCE & BEHAVIOR, infine la voce SYSTEM SETTINGS e deselezioniamo l'opzione REOPEN PROJECT ON STARTUP.

Altre opportunità di configurazione troviamo scegliendo sulla sinistra della finestra la modalità PLUGINS.
Scorrendo l'elenco possiamo scegliere come arricchire il nostro ambiente di sviluppo.
Tra i plugin possiamo scegliere EDU TOOLS, che rende il nostro ambiente simile a quello della versione Edu di Intellij IDEA.

3 Impostazione del progetto

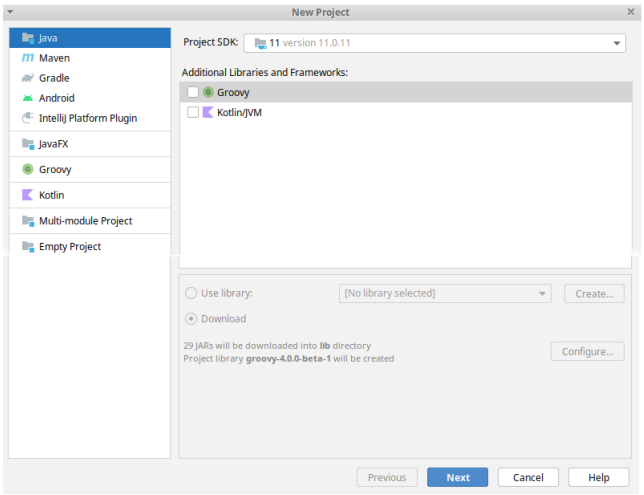
La schermata iniziale riprodotta a pagina 3, oltre alle due modalità viste nel paragrafo precedente, ci propone anche la modalità LEARN INTELLIJ IDEA, che contiene un help e guide introduttive all'uso del software in inglese. A chi conosce questa lingua consiglio di dare un'occhiata in quanto il tutto è fatto molto bene.

Ma per cominciare a programmare dobbiamo scegliere la modalità PROJECTS, che è quella che ci viene proposta per default al lancio.

Come vediamo nella figura di pagina 3, ci vengono proposte tre modalità di avvio del nostro lavoro.

La terza, GET FROM VCS, la lasciamo a chi ne sa più di noi. VCS sta per Version Control System ed è il sistema di controllo delle versioni ideato nel 2005 da Linus Torvalds, l'inventore di Linux, per facilitare lo sviluppo ordinato di software libero ed è gestito da GitHub.

A noi basta la scelta tra NEW PROJECT e OPEN, a seconda se dobbiamo avviare un nuovo progetto o continuare a lavorare su un progetto già avviato. Con NEW PROJECT apriamo questa finestra



Rispetto alla figura precedente questa ha lo sfondo chiaro in quanto, nel frattempo, ho modificato il Color theme da Darcula a Intellij Light.

Di tutto quanto viene proposto sulla sinistra, in questo manualetto ci occuperemo semplicemente di Java e Kotlin e tanto basterà per familiarizzare con Intellij IDEA, in modo da essere poi in grado eventualmente di fare altre cose più difficili.

Ciò di cui ci occuperemo, comunque, non è poco.

4 Programma console in Java

Vediamo ora come scrivere un programmino in linguaggio Java da eseguirsi a terminale, senza GUI.

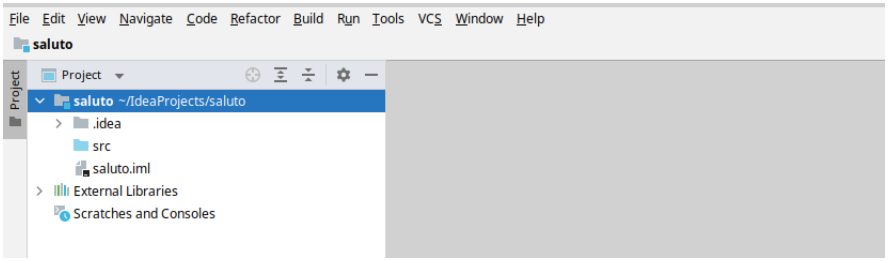
La finestra riprodotta nella pagina precedente, che si apre con NEW PROJECT è forse già predisposta con selezionata la scelta Java. Se così non è scegliamo noi Java.
Previa verifica che nella finestrella in alto, PROJECT SDK, sia indicata la versione del nostro JDK Java, lasciamo così la nuova finestra che si apre, ignorando la proposta scelta di librerie e frameworks addizionali, e clicchiamo su NEXT.

Se non fosse indicata alcuna versione di JDK nella finestrella, agendo sulla finestrella stessa potremmo scegliere l'opzione DOWNLOAD JDK... e procurarcelo.

La finestra che compare dopo aver cliccato su NEXT ci offre la possibilità di avvalerci di un modello di progetto. Per i nostri scopi possiamo ignorare la proposta e cliccare ancora su NEXT.

Nella finestra successiva diamo un nome al progetto, scrivendolo nella finestrella PROJECT NAME, lasciamo inalterato ciò che ci viene proposto nella finestrella PROJECT LOCATION e clicchiamo su FINISH.
Per l'esercizio che propongo andrebbe bene il nome `saluto`.

Dopo qualche secondo il progetto è creato.
Se abbiamo lasciato inalterato il contenuto della finestrella PROJECT LOCATION i file relativi al progetto saranno ospitati in una directory IDEAPROJECTS nella nostra home, esattamente in una sottodirectory nominata con il nome che abbiamo assegnato al progetto.
Sullo schermo compare la finestra del progetto, di cui è qui riprodotta la zona in alto a sinistra



Procediamo cliccando destro su `scr`, la directory destinata a contenere il source code, e nel menu che si apre clicchiamo su NEW e poi su JAVA CLASS.
Diamo quindi un nome alla classe nella finestrella che si apre (nel nostro caso potrebbe essere `Saluto`) e proseguiamo dando INVIO.

A questo punto, nell'ampia zona di destra della finestra del progetto, che è un vero e proprio editor, troviamo i primi elementi per scrivere la nostra classe Java.

Se il nostro programma debba consistere nel chiedere il nome all'utente per salutarlo, dovremmo completare questi primi elementi in modo che nell'editor sia scritto quanto segue:

```
import java.io.*;
public class Saluto {
    public static void main(String[] args) throws Exception {
        BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Come ti chiami?");
        String nome = input.readLine();
        System.out.println("Ciao " + nome + "!");
        System.exit(0);
    }
}
```

Scrivendo il codice ci accorgiamo di come Intellij IDEA ci assista con la code completion.

Possiamo vedere se il nostro codice funziona cliccando destro sulla scritta SALUTO.JAVA che compare sopra l'editor per poi cliccare su RUN 'SALUTO.MAIN()'.

Se tutto funziona il progetto viene compilato, altrimenti veniamo avvisati della presenza di errori.

Indipendentemente da questa prova possiamo compilare il progetto ricorrendo al menu BUILD e scegliendo BUILD PROJECT oppure possiamo semplicemente cliccare sulla piccola icona a forma di martello che troviamo in un paio di posizioni nella finestra del progetto.

Dopo la compilazione, navigando nella sottodirectory `saluto` della directory `IdeaProjects` troviamo il sorgente Saluto.java (in `src`) e il compilato Saluto.class (in `out/production/saluto`).

5 Programma console in Kotlin

Nella finestra che si apre con NEW PROJECT, riprodotta a pagina 4, nell'elenco di sinistra, clicchiamo su Kotlin.

Nella finestra successiva, nella finestrella NAME, scriviamo il nome del progetto (per l'esercizio che propongo suggerisco il nome `saluto_bis`).

Lasciamo inalterato il contenuto della finestrella LOCATION e lasciamo selezionato CONSOLE APPLICATION nella zona PROJECT TEMPLATE.

Nella zona BUILD SYSTEM selezioniamo GRADLE KOTLIN in quanto delle tre alternative proposte mi sembra quella che funzioni meglio.

Proseguiamo con NEXT, accettiamo la finestra come viene proposta e clicchiamo su FINISH.

Si apre così la finestra del progetto e dobbiamo attendere qualche secondo per fare in modo che si completi la configurazione di Gradle, che sarà completata quando nella barra di stato in fondo alla finestra non ci sarà più segno di alcuna attività.

A configurazione completata, cancelliamo il contenuto dell'editor nella finestra di progetto e, per produrre con il linguaggio Kotlin un programma analogo a quello del precedente paragrafo (consistente nel chiedere il nome all'utente per salutarlo), scriviamo questo codice

```
fun main()
{
    println("Come ti chiami?")
    var nome = readLine()
    println("Ciao $nome")
}
```

Anche in questo caso potremo avvalerci della code completion, questa volta riferita al linguaggio Kotlin.

Possiamo vedere se il nostro codice funziona cliccando destro sulla scritta MAIN.KT che compare sopra l'editor per poi cliccare su RUN 'MAINKT'.

Se tutto funziona il progetto viene compilato, altrimenti veniamo avvisati della presenza di errori.

Indipendentemente da questa prova possiamo compilare il progetto ricorrendo al menu BUILD e scegliendo BUILD PROJECT oppure possiamo semplicemente cliccare sulla piccola icona a forma di martello che troviamo in un paio di posizioni nella finestra del progetto.

Dopo la compilazione, navigando nella sottodirectory `saluto_bis` della directory `IdeaProjects` troviamo il sorgente Main.kt (in `src/main/kotlin`) e il compilato MainKt.class (in `build/classes/kotlin/main`).

6 Creazione del file .jar eseguibile

I file compilati che abbiamo creato negli esempi trattati nei due precedenti paragrafi sono entrambi costituiti da bytecode per la Java Virtual Machine.

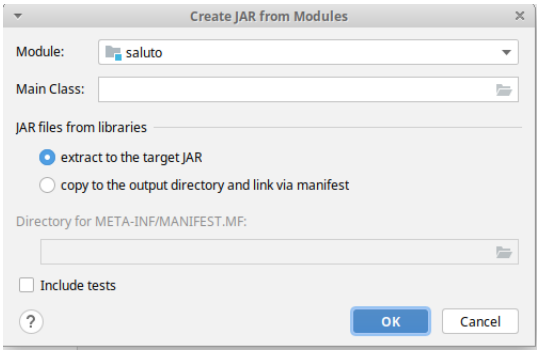
Tuttavia, solo quello prodotto nel paragrafo 4 con il linguaggio Java (Saluto.class) può essere eseguito direttamente con il comando `java Saluto` su qualsiasi computer dotato della JVM.

Quello prodotto nel paragrafo 5 con il linguaggio Kotlin (MainKt.class) può essere eseguito sulla JVM solo passando attraverso una libreria di runtime che fa parte del pacchetto Kotlin e richiede il comando `kotlin MainKt`, cioè occorre che sul computer dove viene eseguito sia presente non solo la JVM ma anche il software Kotlin.

Senza considerare il fatto che se, contrariamente a quanto fatto in modo molto semplice nei nostri esempi, avessimo utilizzato nella programmazione anche librerie esterne a quelle che costituiscono il sistema Java (per esempio predisposte da noi stessi per semplificare certi passaggi della programmazione), anche il programma prodotto in linguaggio Java, per essere eseguito su qualsiasi computer, dovrebbe essere accompagnato dalle librerie esterne a quelle del pacchetto Java che abbiamo utilizzato.

Per ovviare a questi inconvenienti e fare in modo che il programma che produciamo possa essere direttamente eseguito sulla JVM esiste la possibilità di radunare in un solo file tutto ciò che è servito per produrre il programma e, nel caso si tratti di un programma prodotto in linguaggio Kotlin, anche di ciò che serve per utilizzarlo sulla JVM (la libreria di runtime Kotlin): questo unico file, impropriamente chiamato eseguibile, è in realtà un archivio compresso che contiene tutta la raccolta di classi necessarie perché il programma giri direttamente su una Java Virtual Machine. La sua estensione è `.jar` (java archive) e viene utilizzato con il comando
`java -jar` <nome_file>`.jar`
su qualsiasi computer ove sia presente la JVM, indipendentemente dal fatto che sia stato programmato con il linguaggio Java o con il linguaggio Kotlin.

Per produrre questo file con Intellij IDEA, una volta che abbiamo compilato il nostro progetto con il Build, da menu FILE clicchiamo su PROJECT STRUCTURE... e, nella finestra che si apre, clicchiamo su ARTIFACTS nell'elenco del PROJECT SETTINGS.
Clicchiamo ora sul simbolo + sopra la colonna bianca che compare e poi sulla voce JAR del sotto-menu ADD che ci si presenta e scegliamo la voce FROM MODULES WITH DEPENDENCIES...
Si apre così la finestra di creazione del file JAR



Ciò che dobbiamo fare è semplicemente inserire nella finestrella MAIN CLASS il nome della classe (quello del file con estensione `.class`) che, nel nostro progetto, contiene il metodo `main`. Per il nostro esercizio del paragrafo 4 si tratta della classe `Saluto` e per quello del paragrafo 5 si tratta della classe `MainKt`.
Inserito il nome diamo OK e il file archivio jar viene prodotto.

Ora dobbiamo compilarlo e lo facciamo scegliendo da menu BUILD la voce BUILD ARTIFACTS..., cliccando su BUILD nel piccolo menu che compare.

A questo punto nelle directory dove sono ospitati i nostri progetti troviamo una sottodirectory `out` e andando in fondo al percorso lungo la sottodirectory `artifacts` raggiungiamo i file `.jar` che abbiamo prodotto: `saluto.jar` per l'esercizio del paragrafo 4 e `saluto_bis.jar` per l'esercizio del paragrafo 5.

Essi sono parimenti eseguibili con il comando `java -jar` su qualsiasi computer e su qualsiasi sistema operativo sia dotato della Java Virtual Machine, senza più riguardo al fatto che siano stati creati in linguaggio Java o in linguaggio Kotlin.

7 Programmi con GUI

Il linguaggio Java ci offre due strumenti per creare interfacce grafiche per i nostri programmi: il più anziano è la libreria grafica Java AWT (Java Abstract Widget Toolkit), il più recente è la libreria grafica Java Swing.

Il secondo è una evoluzione del primo e offre una strumentazione più ricca; soprattutto, contrariamente al primo, è totalmente svincolato dal sistema operativo su cui viene utilizzato e garantisce così una grafica omogenea degli applicativi, che si presentano senza alcuna differenza sul sistema Windows, su Linux o su Mac.

Più recentemente è nato JavaFX, che consente di produrre una grafica di ineguagliata perfezione, superando il ristretto ambito delle applicazioni desktop per spaziare nel campo delle rich Internet application. Non si tratta più, tuttavia, di una libreria grafica di Java ma si tratta di una vera e propria famiglia di software basata sulla piattaforma Java, né per sviluppare applicazioni JavaFX basta il linguaggio Java.
IntellijIDEA, anche nella versione Community, è dotata di un plugin che consente di lavorare con JavaFX.

In questo manualetto ci limitiamo a ciò che fa strettamente parte del linguaggio Java, più in particolare della libreria grafica Java Swing.

Con Intellij IDEA possiamo utilizzarla richiamandola sia programmando con il linguaggio Java sia programmando con il linguaggio Kotlin, che non ha librerie grafiche sue proprie.

In ogni caso siamo assistiti dalla ricca code completion di Intellij IDEA e possiamo lavorare come abbiamo fatto nei paragrafi 4 e 5 per i programmi console, con l'aggiunta delle istruzioni per produrre la GUI nel momento in cui lavoriamo nell'editor per scrivere il programma.

Il codice per produrre una semplice finestra, base per creare una GUI, scritto in linguaggio Java è il seguente:

```
import javax.swing.JFrame;
public class Frame
{
    public static void main(String[] args)
    {
        JFrame mioFrame = new JFrame("FRAME DI ESEMPIO");
        mioFrame.setBounds(200,100,300,150);
        mioFrame.setVisible(true);
    }
}
```

scritto nel più raccolto e semplice linguaggio Kotlin è il seguente:

```
import javax.swing.JFrame
fun main()
{
    var mioFrame = JFrame("FRAME DI ESEMPIO")
    mioFrame.setBounds(200,100,300,150)
    mioFrame.setVisible(true)
}
```

In ogni caso viene visualizzata questa finestra in alto sulla sinistra dello schermo, a 100 pixel dall'alto e a 200 pixel da sinistra

8 Swing GUI designer

La scrittura del codice per creare un'interfaccia grafica di una certa complessità richiede un bel po' di lavoro, soprattutto avendo a che fare con il verboso e complicato linguaggio Java.

Né possiamo trovare sollievo utilizzando il più semplice linguaggio Kotlin che, per la grafica, deve utilizzare le librerie Java con il relativo linguaggio.

Intellij IDEA ci viene incontro con lo strumento del GUI designer, con il quale possiamo creare l'interfaccia grafica disegnandola visualmente. In questo modo siamo sollevati dal compito di scrivere il codice per creare l'interfaccia e ci resta solo da scrivere il codice per gestire i collegati eventi.

Il designer di Intellij IDEA non è il massimo ed è un tantino complicato da usare: è molto migliore quello che ci offre NetBeans, altro famoso IDE per Java. Inoltre, non si capisce per quale motivo, esso ha una strana limitazione: non ci dà modo di creare menu.

Come accade con tutti gli strumenti visuali, ci aiuta l'intuito e la pratica.

Per cominciare a famigliarizzare propongo il seguente esercizio, nel quale creeremo il solito programma che chiede all'utente il nome per salutarlo, questa volta con interfaccia grafica.

Nella finestra che si apre con NEW PROJECT, riprodotta a pagina 4, nell'elenco di sinistra, clicchiamo su JAVA.

Previa verifica che nella finestrella in alto, PROJECT SDK, sia indicata la versione del nostro JDK Java, lasciamo così la nuova finestra che si apre, ignorando la proposta scelta di librerie e frameworks addizionali, e clicchiamo su NEXT.

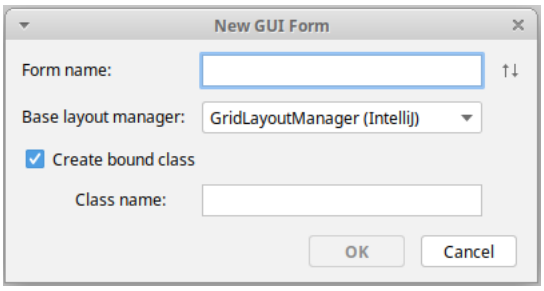
Altrettanto ignoriamo la possibilità che ci viene offerta nella finestra successiva di avvalerci di un modello di progetto e clicchiamo ancora su NEXT.

Nella nuova finestra inseriamo il nome del progetto nella finestrella PROJECT NAME, lasciamo inalterato ciò che ci viene proposto nella finestrella PROJECT LOCATION e clicchiamo su FINISH.

Per l'esercizio che propongo andrebbe bene il nome `saluto_gui`.

Procediamo cliccando destro su `scr`, la directory destinata a contenere il source code, e nel menu che si apre clicchiamo su NEW, poi su SWING UI DESIGNER e quindi su GUI FORM.

Ci si presenta questa finestra di dialogo



nella cui finestrella FORM NAME dobbiamo scrivere un nome da dare al Form. Nel nostro caso suggerisco `SalutoGUI`. Esso viene automaticamente ripetuto nella finestrella CLASS NAME e va bene così.

Clicchiamo su OK e arriviamo finalmente nella finestra in cui lavorare al nostro progetto, riprodotta nella pagina seguente.

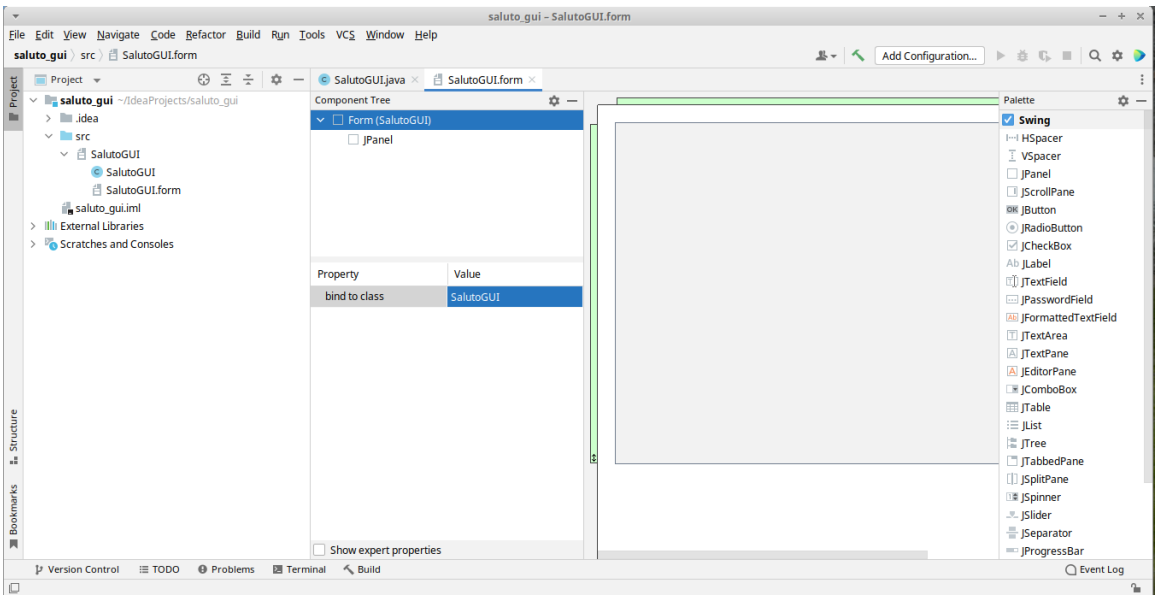
In alto abbiamo la solita barra dei menu, identica a quella della finestra in cui costruiamo programmi console.

Sotto, sulla sinistra, abbiamo la solita zona di navigazione nelle directory e nei file del progetto.

Appena a destra abbiamo la zona di navigazione tra i widget.

Segue la zona in cui disegnamo la GUI e, infine, sull'estrema destra, abbiamo l'elenco dei widget disponibili per disegnare la GUI.

Widget (abbreviazione di window gadget) è il termine, comune a tutti i sistemi di costruzione di interfacce grafiche, che identifica i vari componenti dell'interfaccia stessa: pannelli, pulsanti, scritte, finestrelle per l'inserimento di dati, ecc.

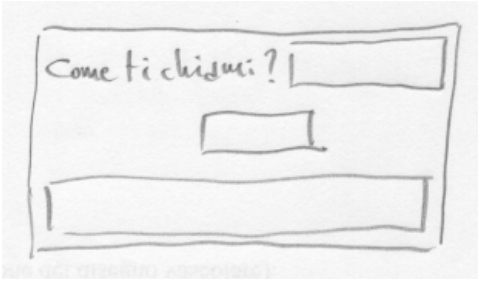


Nella zona di navigazione tra i widget notiamo che automaticamente Intellij IDEA ha già aggiunto al form un widget contenitore JPanel che occupa tutto il form.
Lo selezioniamo con un click del mouse e così apriamo la finestra delle proprietà, nella quale, in corrispondenza alla PROPERTY `field name`, nella casella VALUE, inseriamo un nome: suggerisco `base`.

Ora clicchiamo sulla tacca, sopra l'editor, SALUTOGUI.JAVA per verificare che, oltre alla parte del disegno, si sia creato anche il codice corrispondente, cioè la classe contenente il primo widget e vi inseriamo un metodo `main` per fare in modo che l'applicazione poi giri.
Per farlo posizioniamo il mouse appena dopo il punto e virgola dell'unica riga contenuta nel codice della classe, diamo INVIO e da menu CODE ▷ GENERATE scegliamo FORM MAIN().
Possiamo lanciare l'applicazione con click destro sulla tacca SALUTOGUI.JAVA scegliendo RUN 'SALUTOGUI.MAIN()'. L'applicazione partirà, tuttavia senza farci vedere nulla in quanto non c'è ancora nulla da vedere.

Prima di proseguire dobbiamo avere le idee chiare su ciò che vogliamo fare e, a questo scopo, è bene che facciamo uno schizzo della GUI che intendiamo realizzare.

Nel nostro caso penserei ad una cosa così



In alto una zona per l'input, con la richiesta di cosa occorre immettere affiancata da un widget che consenta l'immissione: nella terminologia di Java Swing ci serviranno un widget JLabel affiancato da un widget JTextField.
Sotto possiamo collocare un pulsante per eseguire il saluto una volta che sarà stato scritto il nome della persona da salutare: qui ci servirà un JButton.
Sotto ancora una zona per l'output, dove venga scritto il saluto: ci servirà un'altra JLabel.

Per inserire i widget che ci servono occorre un widget contenitore e uno strumento per gestire la geometria dell'inserimento: nella terminologia di Java Swing, un JPanel e un Layout manager.

Attualmente il nostro progetto ha un JPanel ma il Layout manager è un GridLayout, cioè gestisce gli inserimenti a griglia e, pertanto, non saremo mai in grado di ottenere un allineamen-

to centralizzato di un widget sotto una riga che ha già inizializzato due colonne della griglia (purtroppo il designer di Intellij IDEA non contempla il raggruppamento di celle).

Come spesso occorre fare in questi casi dobbiamo utilizzare diversi JPanel, ciascuno con un Layout manager adatto per ciò che vogliamo fare.

Nel nostro caso cominciamo con un dimensionamento di massima del Form che ci viene proposto con una dimensione di molto superiore a quella di cui abbiamo bisogno.
A questo scopo clicchiamo sul form e vediamo che, in realtà, abbiamo selezionato il widget JPanel che lo copre integralmente.
Ora, agendo con il mouse e il suo trascinamento sulle maniglie che contornano il pannello nella zona di disegno, riduciamo il tutto (Form e Pannello sovrastante) a dimensioni coerenti con il nostro schizzo.
Possiamo spostare a dovere il risultato del nostro ridisegno agendo sulla maniglia di spostamento che si apre appena fuori dall'angolo in alto a sinistra del form.

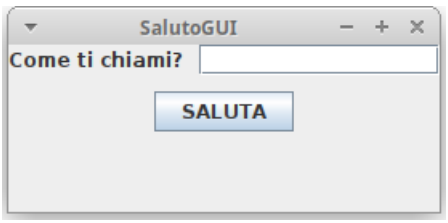
Avviamo quindi, finalmente, la costruzione della nostra GUI.

Cominciamo con la sovrapposizione all'unico JPanel che abbiamo chiamato `base` di un altro JPanel, nella parte alta del form, che possiamo gestire con il Layout manager di default.
A questo scopo selezioniamo nella barra verticale il widget JPanel e lo trasciniamo con il mouse sul JPanel `base`, in alto.
Nella finestra delle Proprietà gli attribuiamo un nome: per esempio `alto` e lasciamo inalterata la scelta del Layout manager.
Vi inseriamo, sempre per trascinamento del mouse, un widget JLabel e un widget JTextField.
Il primo lo nominiamo `richiesta` e, scorrendo le sue proprietà, inseriamo in corrispondenza della proprietà TEXT la dicitura `Come ti chiami?`
Il secondo lo nominiamo `input`.

Ora inseriamo un altro JPanel per ospitare il pulsante centrato nel form e, al fine di poter avere un pulsante di dimensione proporzionata alla scritta che contiene e non largo come tutta la finestra, lo gestiamo con il FlowLayout.
A questo scopo trasciniamo un widget JPanel collocandolo appena sotto il precedente, lo nominiamo `medio` e, nelle proprietà, modifichiamo il Layout manager in FLOWLAYOUT.
Vi trasciniamo un widget JButton, che chiamiamo `esegui`, e, in corrispondenza alla proprietà TEXT, inseriamo la dicitura `SALUTA`.

Inseriamo infine un terzo JPanel, immediatamente sotto al precedente, lo nominiamo `basso`, e manteniamo il Layout manager di default.
Vi inseriamo un widget JLabel che chiamiamo output.
Settiamo la sua proprietà HORIZONTAL ALIGN in `Center` e la proprietà HORIZONTALALIGNMENT in `Center`.
Apriamo la proprietà PREFERRED SIZE e in corrispondenza di HEIGHT inseriamo il VALUE `40`, in modo da assicurarci spazio sufficiente per scrivere il saluto.
Eliminiamo il contenuto della casella VALUE in corrispondenza della proprietà TEXT.

Possiamo lanciare l'applicazione con click destro sulla tacca SALUTOGUI.JAVA scegliendo RUN 'SALUTOGUI.MAIN()'. Se tutto è andato a dovere vediamo questa finestra



Ci manca di far funzionare il tutto, cioè di ottenere che, inserito un nome nella finestrella e premuto il pulsante `SALUTA`, compaia il saluto.

Ma è presto fatto.

Clicchiamo destro sul pulsante `SALUTA` nella zona disegno, scegliamo dal menu CREATE LISTENER e, successivamente, ACTIONLISTENER.

Data conferma alla successiva finestra, nella finestra del codice della classe SALUTOGUI.JAVA ci si predispone il codice per gestire l'evento consistente nel click sul pulsante e un cursore lampeggiante ci invita a completare ciò che manca.
Lo facciamo in linguaggio Java inserendo quanto segue

```
String saluto = "Ciao " + input.getText();
output.setText(saluto);
```

Lanciata nuovamente l'applicazione, inserito il nome della persona da salutare e premuto INVIO, la nostra GUI compare così

Per chiudere l'applicazione clicchiamo sul simbolo `x` in alto a destra.

Con il procedimento visto nel Paragrafo 6 possiamo creare un eseguibile `.jar` e distribuire la nostra applicazione, che girerà sulla JVM.

* * *

Dal momento che la classe Java che genera la GUI, dotata del metodo `main` in linguaggio Java, è già un programma inseribile in un eseguibile `.jar`, il trascinare l'utilizzo di questa classe in un metodo `main` di Kotlin per arrivare allo stesso risultato potrebbe apparire un esercizio inutile. A meno che il programma Kotlin non abbia la sola finalità di utilizzare la classe Java per la GUI ma abbia anche altre finalità.

Per chi voglia fare l'apparentemente inutile esercizio o per chi abbia in mente cose più complesse accenno al modo di procedere.

In Intellij Idea si crea un nuovo progetto con NEW PROJECT, si sceglie JAVA e si clicca sull'opzione KOTLIN/JVM nella zona ADDITIONAL LIBRARIES AND FRAMEWORKS.

Nella successiva finestra si dà un nome al progetto e si clicca su NEXT.

Nella finestra del progetto clicchiamo destro sulla cartella `Src`, scegliamo NEW nel menu che compare e con questo sistema creiamo prima un KOTLIN FILE/CLASS, scegliendo FILE e poi un SWING UI DESIGNER, scegliendo GUI FORM, dando loro nomi appropriati.

Appena creato il GUI Form vediamo comparire nella cartella Src e nella barra sovrastante l'editor del progetto il file Kotlin con estensione `.kt` e i due file della GUI, uno per il codice e uno per il disegno e ci ritroviamo aperto quello per il disegno.

Disegnamo la GUI avvalendoci del designer secondo il procedimento visto prima.

Una volta disegnata la GUI, apriamo il file di codice Java generato dal designer, cliccando sulla relativa tacca sopra l'editor, e copiamo il contenuto che c'è tra le parentesi graffe della funzione `main`.
Inoltre, dal momento che il designer ha generato il pannello come private, dobbiamo renderlo public cliccando destro appena dopo la parentesi graffa di chiusura della funzione `main`, scegliendo GENERATE e poi GETTER ed il pannello nell'elenco che compare.

Quindi apriamo il file per il codice `.kt`, cliccando sulla relativa tacca sopra l'editor, creiamo la funzione `main()` e all'interno delle parentesi graffe inseriamo quanto avevamo copiato prima, dando il nostro assenso alla richiesta di conversione che ci viene posta.

Compiliamo il file `.kt` ed abbiamo il nostro programma Kotlin dotato di GUI.